UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA DE TELECOMUNICACIÓN

PLANIFICACIÓN Y GESTIÓN DE TELECOMUNICACIONES

PROYECTO FIN DE CARRERA

CONTROL DE SENSORES ELECTRO-ÓPTICOS EN AVIONES NO TRIPULADOS Y TRATAMIENTO DE IMÁGENES UTILIZANDO MÁQUINAS DE VECTORES SOPORTE

Autora: MARÍA ISABEL ESCOBAR ESCALERO
Tutor: MARCELINO LÁZARO
Director: JOSÉ ÁNGEL GINER LILLO

JUNIO DE 2010

Proyecto Fin de Carrera

CONTROL DE SENSORES ELECTRO-ÓPTICOS EN AVIONES NO TRIPULADOS Y TRATAMIENTO DE IMÁGENES UTILIZANDO MÁQUINAS DE VECTORES SOPORTE

Autora

MARÍA ISABEL ESCOBAR ESCALERO

Directores

MARCELINO LÁZARO (UC3M)

JOSÉ ÁNGEL GINER LILLO (EADS)

La defensa del presente Proyecto Fin de Carrera se realizó el día 10 de Junio de 2010, siendo calificada por el siguiente tribunal:

PRESIDENTE: Francisco Javier González Serrano

SECRETARIO: Jerónimo Arenas García

VOCAL: Patricia Arias Cabarcos

y habiendo obtenido la siguiente calificación:

CALIFICACIÓN: MATRÍCULA DE HONOR

Leganés, a 10 de Junio de 2010

❝La gente que se arriesga y lucha contra sus miedos,

CONSIGUE LO QUE QUIERE.❞

Agradecimientos

En un principio no iba a comenzar esta andadura que hoy pone fin a una larga cantidad de noches en vela y horas de estudios, alegrías y penas, compartidas con familiares y amigos. Por todo ello, por formar parte de esta gran andadura, os merecéis este pequeño homenaje.

Marce y Giner, ¿qué hubiera hecho sin vosotros? Gracias por todo lo que me habéis enseñado, por las horas que me habéis dedicado, por confiar en mí y ver siempre, que era capaz de sacar el proyecto adelante. ¡Mil gracias!

Compañeros de trabajo, quien me iba a decir hace un año, cuando di el paso de formar parte de esta gran aventura que hoy ibais a estar conmigo compartiendo este momento. Gracias por las celebraciones, por las risas, por los momentos compartidos y, sobre todo, por el gran apoyo que me habéis brindado en todo momento!

Ana, Esther, Hodei y Piero, me regalasteis el mejor año de mi vida, lo compartisteis conmigo, y desde entonces seguimos compartiendo nuestras vidas. Han pasado ya casi cuatro años cuando nos embarcamos en la gran aventura del Erasmus. Parece que fue ayer el día que nos conocimos, como todo en esta vida, por casualidad, pero eso es lo bonito, ¿no? Nunca os olvidaré. ¡Os quiero "familia"! ¡Gracias a todos los que formaron parte de mi gran sueño erasmus!

María y Patricia, gracias por hacer de los laboratorios una aventura, por darme tan buenos momentos. Hoy consigo el sueño común que hemos compartido como compañeras de universidad y como amigas. Recordad siempre que "NUSOTRAS PUDEMOS". ¡Os quiero guapas!

Juanfran, gracias por las charlas interminables, por el apoyo incondicional, por los consejos, por las noches en vela. ¡Te adoro!

Diego, nos conocimos hace ya cinco años, comenzamos siendo compañeros y hoy somos amigos, mil gracias por los consejos, los ánimos, las celebraciones, por estar siempre ahí, por saberlo todo sin necesidad de decir nada, por todo. ¡Te quiero un montón!

Alberto, nos conocimos en la EOI, cuando para sobrevivir a la presión de la universidad nos "divertíamos" estudiando inglés, luego resultó que éramos vecinos y ahora amigos, gracias por el apoyo que siempre me has brindado, por alegrarme con cada charla, cada llamada. ¡Mr. Cobra forever!

Marta, me has sufrido desde hace ya diez años, casi, desde que comenzó mi aventura universitaria. Creo que sobran las palabras, lo sabes todo "hermana"! ¡Te quiero!

Mari, ¿te acuerdas hace ya casi veinticuatro años cuando nos presentaron nuestros padres, cuando corríamos por el colegio al tocar la sirena? ¡quién iba a decirnos que íbamos a vivir tantas cosas, qué después de todo este tiempo íbamos a ser como hermanas! ¿qué le voy a decir a mi "hermana" que no sepa? ¡Te quiero!

Tios y primos, mil gracias por ser parte de esta maravillosa familia, por ayudarme cada día a ser la persona que soy, por estar siempre ahí. ¡Os quiero!

A mis abuelos les quiero dedicar una mención especial, desde pequeñita me habéis enseñado y posteriormente guiado desde ahí arriba para que consiga mis sueños, para que ante las adversidades consiga levantarme y celebre los triunfos. Isabel, Milagros y Ángel os llevo en mi corazón en cada paso del largo camino de la vida. Yayo Atilano, ¿cuántas veces me habrás dicho...cuándo terminarás de estudiar preciosa? Hoy te puedo decir con orgullo de nieta que he terminado y ¡SOY INGENIERA!

Mamá, Papá y Goyi, sois mi vida, hay una canción que se llama “Estoy hecho de pedacitos de ti”, pues eso soy yo, un cachito de cada uno de vosotros. No me cansaré nunca de daros las gracias, cuánto hemos luchado hasta llegar al día de hoy y sintiéndolo mucho, pero lo que nos queda por luchar! La verdad es que hacemos un gran equipo, no es porque sea el mío, pero es un equipo ganador en todos los sentidos. Juntos hemos celebrado los aprobados, hemos llorado los suspensos, hemos plantado cara a las adversidades y nos hemos enorgullecido de los triunfos, nos hemos apoyado y cuidado, y hoy, a los cuatro, nos toca celebrar que soy ¡INGENIERA! Como os he dicho, ¡SOIS MI VIDA! ¡OS QUIERO!

Para finalizar, me gustaría deciros que este título es también vuestro, hoy os podéis sentir todos Ingenieros. Mil gracias a todos por haber contribuido a hacer este sueño . . .

. . . ¡UNA REALIDAD!

María Isabel Escobar Escalero.

Resumen

Hoy en día, la revolución tecnológica, permite realizar cosas que hasta hace unos años eran impensables; como por ejemplo, identificar personas por los rasgos faciales o compartir gran cantidad de información a través de una red llamada internet.

El sector militar aeronáutico es uno de los que más invierte en investigación, desarrollo e implantación de los más nuevos y sofisticados avances tecnológicos y que en el futuro se aplicarán en el sector civil.

Actualmente, el boom aeronáutico pasa por el desarrollo de aviones no tripulados, los llamados UAVs (del inglés, *Unmanned Aerial Vehicles*). Las empresas punteras del sector, como EADS (*European Aeronautics Defence and Space Company*) o Boeing, están invirtiendo en el desarrollo de UAVs cada vez más sofisticados; capaces de realizar misiones de mayor duración y con la tecnología más puntera, con el objetivo de llevar a cabo las misiones requeridas con total éxito.

Este proyecto se ha planteado dentro de la empresa EADS, gracias a los nuevos desarrollos que se están llevando a cabo en la división de *Defence & Security*. Este proyecto, surgió a raíz del nuevo desarrollo llamado UAV ATLANTE (Avión Táctico de Largo Alcance No Tripulado) el primer UAV español, el cual está desarrollando para el ejército español la empresa EADS. Entre las misiones más importantes que desarrollan los UAVs se encuentran las misiones de inteligencia, vigilancia, adquisición de blancos y reconocimiento. Para ello, el avión lleva incorporado en la parte inferior del fuselaje un sensor EO (Electro-Óptico), para el cual se va a desarrollar este proyecto.

A pesar de que el proyecto surgiera a raíz del desarrollo del UAV ATLANTE, éste es aplicable a cualquier UAV que esté provisto de un sensor EO, en cuyos computadores o en los de la estación base, sea posible integrar los algoritmos desarrollados.

El proyecto se ha subdividido en dos subproyectos de disciplinas muy dispares, que han sido aplicados de forma combinada para lograr un fin común.

El primer subproyecto, desarrollado en la empresa EADS, consiste en la implementación de los algoritmos de geo-apuntamiento (*geopointing*) y geo-localización (*geolocation*). El algoritmo de geo-apuntamiento permite calcular los ángulos del sensor EO para que éste apunte a una latitud, longitud y altura determinadas por el operador del UAV situado en tierra. El algoritmo de geo-localización permite calcular la latitud, longitud y altura de la posición geográfica a la que la cámara apunte en el momento que el operador le indique al UAV.

Una vez implementados los dos algoritmos, se realizará una batería de pruebas para verificar el correcto funcionamiento de los mismos. Las pruebas se realizarán en primer lugar en un PC, así se eliminarán posibles fallos de codificación, y a continuación, se realizará una integración de los algoritmos en un entorno de simulación. Esta integración se llevará a cabo en un banco de pruebas perteneciente a EADS. Las pruebas de integración en banco permitirán comprobar el correcto funcionamiento de los algoritmos de manera visual.

La cámara del sensor EO del UAV es capaz de capturar imágenes en tiempo real. El procesado y análisis de estas imágenes para la detección de posibles objetivos como, fuegos, personas, convoyes de misión humanitaria... son fundamentales para el correcto desarrollo de las misiones de un UAV.

En la segunda parte del proyecto, la cual ha sido desarrollada en la UC3M (Universidad Carlos III de Madrid), se quiere realizar una implementación de un clasificador de imágenes, y discutir en torno a la viabilidad de embarcar la implementación en el UAV o ejecutarla en la estación de

tierra. Para su realización se ha optado por utilizar una implementación de las máquinas de vectores soporte (SVM, *Support Vector Machines*); ya que permiten realizar una clasificación máquina con un alto porcentaje de aciertos, a la vez que se minimiza la cantidad de información necesaria.

Abstract

Nowadays, technological revolution has allowed to develop features absolutely unbelievable some years ago. For example, identifying people by facial features or sharing a wealth of information through a network called the Internet.

The military aircraft sector is one of the largest investors in research, development and implementation on the newest and most sophisticated technological advances which will be applied in the civilian sector on the future.

Currently, the aviation boom goes through the development of UAVs (Unmanned Aerial Vehicles). Leader companies, such as EADS (European Aeronautics Defence and Space Company) and Boeing, are investing in the development of increasingly sophisticated UAVs, capable to perform longer missions with the leading technology, with the aim to obtain resounding successes on the required missions.

This project has arisen within the company EADS, thanks to new developments that are being developed in the division of Defence & Security. This project arose, because of the new development called UAV ATLANTE (spanish long range tactical aircraft unmanned) the first Spanish UAV, which is developed for the Spanish army by EADS. Among the most important missions that are being developed by UAVs are the missions of intelligence, surveillance, target acquisition and recognition. For this, the aircraft carries incorporated in the belly a EO (Electro-Optical) sensor, for which it will develop this project.

In spite of the fact that the project arose because of the development of the UAV ATLANTE, this is applicable to any UAV that be provided of a EO sensor, in which whether on-board computers or those of the base station, could be possible to integrate in the developed algorithms.

The project has been subdivided in two very different subproyects that have been applied in combination to achieve a common goal.

The first subproject developed at EADS is the implementation of the algorithms for geopointing and geolocation. The geopointing algorithm works out the angles of the EO sensor pointing to the latitude, longitude and height determined by the UAV operator located on land. The geolocation algorithm works out the latitude, longitude and height of the geographic position where the camera is focusing at the moment that the user indicates to the UAV.

Once implemented the two algorithms, a battery of tests will be performed to verify proper operation thereof. The tests will be carry out first on a PC, to remove potential coding errors. Then there will be an integration of algorithms in a simulation environment. This integration will be carry out in a test bench belonging to EADS. The integration test bench will allow testing the proper functioning of the algorithms in a visual way.

The camera of the EO sensor of the UAV is capable of capturing images in real time. Processing and analysing these images to detect possible targets such as fires, people who need rescuing, convoys of humanitarian tasks,... are essential for the correct development of the missions of an UAV.

In the second part of the project, which has been developed in the UC3M (Carlos III University of Madrid), an implementation of an image classifier has been done, and also discussion about the feasibility of deployment on board the UAV or run in the ground station. To achieve this target, an implementation of Support Vector Machines (SVMs) has been chosen, as they allow a classification machine with a high percentage of hits, while minimizing the amount of necessary information.

Índice General

Agradecimientos v

Resumen vii

Abstract ix

Lista de Figuras xv

Lista de Tablas xix

I PRELIMINARES 1

I.1 Introducción **3**
I.1.1 Planteamiento y motivación del proyecto 4
I.1.2 Objetivos 7
I.1.3 Visión general del documento 8
I.1.4 Consideraciones del proyecto 10

I.2 Antedecentes **11**
I.2.1 Sistemas aéreos no tripulados (UAS) 12
I.2.2 Los aviones no tripulados (UAVs) 15
I.2.2.1 Características y misiones que desarrollan 15
I.2.2.2 Clasificación 17
I.2.2.3 Los UAVS de EADS 18
I.2.2.4 El ATLANTE (Avión Táctico de Largo Alcance No Tripulado Español) . . 20
I.2.2.4.1 Diseño del ATLANTE 20
I.2.3 Nociones básicas de cartografía 22
I.2.3.1 Cálculo de la latitud paramétrica (φ_{PARAM}) 24
I.2.3.2 Cálculo de la latitud geodética (φ_{GEOD}) 25
I.2.4 Teoría del tratamiento digital de información 27
I.2.5 Máquinas de vectores soporte (SVMs) 29
I.2.5.1 Caso linealmente separable 30
I.2.5.2 Caso no linealmente separable 33
I.2.5.3 SVM no lineales 34
I.2.5.3.1 La función núcleo $k(x, x_i)$ 36
I.2.5.4 Fortalezas y debilidades de las SVMs frente a las redes neuronales 36

II CONTROL DE SENSORES EO EN UAVS 39

II.1 Sistemas de Referencia **43**
II.1.1 Elipsoide de referencia WGS84 44

II.1.2 Sistema de coordenadas Geodéticas o LLA . . . 46
II.1.3 Sistema de coordenadas ECEF . . . 46
II.1.4 Sistema de coordenadas NED . . . 47
II.1.5 Sistema de coordenadas BODY . . . 48
II.1.5.1 Actitud del avión . . . 48
II.1.6 Matrices de cambio de coordenadas entre sistemas de referencia . . . 49
II.1.6.1 Matrices de rotación . . . 49
II.1.6.1.1 Giro en $\vec{x}$ . . . 49
II.1.6.1.2 Giro en $\vec{y}$ . . . 50
II.1.6.1.3 Giro en $\vec{z}$ . . . 50
II.1.6.2 LLA $\longleftrightarrow$ ECEF . . . 51
II.1.6.3 ECEF $\longleftrightarrow$ NED . . . 52
II.1.6.4 NED $\longleftrightarrow$ BODY . . . 52
II.1.7 Coordenadas y matrices homogéneas . . . 54

II.2 Algoritmos para el control de los sensores EO 57
II.2.1 Caracterización del sensor EO del UAV . . . 58
II.2.1.1 Cálculo de los ángulos Azimut y Elevación a partir de los ejes BODY del avión . . . 58
II.2.2 Algoritmo para el cálculo de la posición de un objetivo (Geo-localización) . . . 60
II.2.3 Algoritmo para el apuntamiento del sensor EO a un objetivo (geo-apuntamiento) 63

II.3 Proyección cónica conforme de Lambert 65
II.3.1 Transformación directa para elipsoides . . . 69
II.3.2 Transformación indirecta para elipsoides . . . 70

II.4 El módulo FSUIPC 71
II.4.1 FS-Interrogate . . . 72
II.4.2 Utilización del módulo FSUIPC . . . 74

II.5 Plan de pruebas 79
II.5.1 Pruebas en PC . . . 80
II.5.1.1 Prueba 1: Cambio de coordenadas LLA $\leftrightarrow$ ECEF . . . 80
II.5.1.2 Prueba 2: Matrices de cambio de coordenadas . . . 80
II.5.1.3 Prueba 3: Cálculo de la actitud de la cámara . . . 81
II.5.1.4 Los algoritmos de geo-apuntamiento y geo-localización . . . 82
II.5.1.4.1 Prueba 4: Algoritmo de geo-apuntamiento . . . 82
II.5.1.4.2 Prueba 5: Algoritmo de geo-localización . . . 82
II.5.1.4.3 Prueba 6: Geo-apuntamiento $\rightarrow$ Geo-localización $\rightarrow$ Geo-apuntamiento 82
II.5.1.4.4 Prueba 7: Barrido en todo el rango de posibles valores de las entradas 83
II.5.2 Pruebas de integración en el banco de Aviónica . . . 84
II.5.2.1 Pruebas del algoritmo de geo-apuntamiento . . . 86
II.5.2.1.1 Prueba 8: Posicionamiento del avión en Torre Eiffel . . . 86
II.5.2.1.2 Prueba 9: Apuntamiento a los objetivos realizando círculos alrededor de ellos. . . . 86
II.5.2.2 Prueba 10: Algoritmo de geo-localización . . . 88
II.5.2.3 Prueba 11: Rendimiento del algoritmo e interacción con el módulo FSUIPC 88

III TRATAMIENTO DIGITAL DE IMÁGENES UTILIZANDO SVMS . . . 91

III.1 Extracción de características (PCA) 95

III.2 Algoritmo *Complex-Log Mapping* (CLM) 99
III.2.1 Logaritmo Complejo . . . 100
III.2.2 Transformada de Fourier (FFT, *Fast Fourier Transform*) . . . 103

III.3 Clasificadores multiclase 105
III.3.1 Uno vs otros - *One vs All* (OvA) . . . 106
III.3.2 Uno único vs Todos - *Unique One vs All* . . . 108
III.3.3 Todos vs Todos - *All vs All (AvA)* . . . 108

III.4 Imágenes empleadas (COIL-100) 111
III.4.1 *Columbia Object Image Library* (COIL)-100 . . . 112
III.4.2 Procesado de las imágenes . . . 115

III.5 Clasificadores de imágenes mediante SVM 117
III.5.1 Librería LIBSVM . . . 118
III.5.2 Arquitectura . . . 120
III.5.3 *Complex-Log Mapping* . . . 122
III.5.3.1 Implementación de la transformación *Complex-Log* . . . 122
III.5.3.2 Interpolación . . . 124
III.5.4 Extracción de características . . . 126
III.5.5 Elección de la función núcleo y sus parámetros . . . 126

III.6 Análisis de resultados 129
III.6.1 Clasificación de imágenes en crudo . . . 132
III.6.1.1 Clasificador binario . . . 132
III.6.1.2 Clasificador multiclase . . . 134
III.6.2 Clasificación de imágenes con ruido . . . 137
III.6.3 Clasificación de imágenes rotadas . . . 139
III.6.3.1 Clasificador binario [*Complex-Log* sin FFT (Fase 1)] . . . 140
III.6.3.2 Clasificador binario [*Complex-Log* con FFT] . . . 142

IV UN PROYECTO AERONÁUTICO. . . . 147

IV.1 Ciclo de vida de un proyecto de software aeronáutico 149

IV.2 Integración a nivel de componente software 155
IV.2.1 Notación asintótica . . . 158
IV.2.1.1 Notación O . . . 158
IV.2.1.1.1 Órdenes de magnitud . . . 158
IV.2.1.2 Notación Ω . . . 159
IV.2.1.3 Notación Θ . . . 160
IV.2.1.4 Reglas de simplificación . . . 160
IV.2.1.5 Ejemplo 1: Sumatorio de una serie aritmética . . . 160
IV.2.1.6 Ejemplo 2: Producto de matrices . . . 161
IV.2.2 Análisis temporal de los algoritmos . . . 162
IV.2.2.1 Control de sensores EO . . . 162
IV.2.2.1.1 Geo-localización . . . 162
IV.2.2.1.2 Geo-apuntamiento . . . 164
IV.2.2.2 Clasificador de imágenes . . . 165
IV.2.3 Análisis espacial de los algoritmos . . . 168
IV.2.4 Conclusiones . . . 169

IV.3 Integración a nivel de sistemas **171**

V CONCLUSIONES Y TRABAJOS FUTUROS 177

V.1 Conclusiones **179**

V.2 Trabajos Futuros **183**

VI APÉNDICES 187

A Presupuesto **189**
A.1 Tareas realizadas durante el proyecto 190
A.2 Análisis y valoración de recursos/costes 192
A.2.1 Costes directos 192
A.2.1.1 Material 192
A.2.1.2 Recursos Humanos 193
A.2.2 Costes indirectos 193
A.2.3 Costes totales 194

B Acrónimos **195**

Bibliografía **199**

Lista de Figuras

I.1.1 Planteamiento inicial del proyecto. 6

I.2.1 Ejemplo de plataforma aérea, UAV ATLANTE [EAD10]. 12
I.2.2 Ejemplo de carga útil, sensor EO [Sen10]. 13
I.2.3 Ejemplo de estación de tierra. 13
I.2.4 Ejemplos de sistemas de lanzamiento y recuperación. 13
I.2.5 Ejemplo de escenario de un sistema UAS. 14
I.2.6 Proyectos de UAVs en los que participa EADS [[EAD10] y [Cor09]]. 19
I.2.7 Prototipos del UAV Atlante [[EAD10] y [Cor09]]. 21
I.2.8 Ejemplo de representación de un punto P sobre la superficie terrestre mediante la latitud y la longitud. 22
I.2.9 Latitudes geocéntrica, paramétrica y geodética en el plano meridional. 23
I.2.10 Latitudes geocéntrica, paramétrica y geodética en el plano meridional. 24
I.2.11 Latitudes geocéntrica, paramétrica y geodética de un punto de altura h. 26
I.2.12 Ejemplos de los efectos del sobreajuste y subajustes [svm10]. 29
I.2.13 Ejemplos de diferentes hiperplanos. 30
I.2.14 Ejemplo de maximización del margen mediante la elección del OSH. 31
I.2.15 Ejemplo de clasificación binaria [SBS98]. 33
I.2.16 Ejemplo de clasificación no linealmente separable. 34
I.2.17 Transformación del espacio de entrada. 35
18 Ejemplos de sensores EO [Sen10]. 41

II.1.1 Sistema Geodésico [Los ejes WGS84 son X_{WGS84}, Y_{WGS84} y Z_{WGS84}]. 45
II.1.2 Sistema de coordenadas LLA. 46
II.1.3 Sistema de coordenadas ECEF. 46
II.1.4 Sistema de coordenadas NED. 47
II.1.5 Sistema de coordenadas BODY. 48
II.1.6 Ángulos del sistema de coordenadas BODY. 48
II.1.7 Sistema de coordenadas realizando un giro en el eje x. 49
II.1.8 Sistema de coordenadas realizando un giro en el eje y. 50
II.1.9 Sistema de coordenadas realizando un giro en el eje z. 51

II.2.1 Sistema de coordenadas LOS. 58
II.2.2 Algoritmo de geo-localización. 60
II.2.3 Cálculo de la posición de un TGT (Geo-localización). 61
II.2.4 Algoritmo de geo-apuntamiento. 63

II.3.1 Ejemplo de una proyección. 66
II.3.2 Ejemplos de proyecciones cilíndricas, cónica y azimutal. 67
II.3.3 Proyección cónica conforme de Lambert. 68

II.4.1 Propiedades de la librería FSUIPC. 72

II.4.2 Valores de las propiedades de la librería FSUIPC. 73
II.4.3 Estructura necesaria para la utilización de FSUIPC. 74
II.4.4 Información de las variables del módulo FSUIPC que se van a utilizar. 76

II.5.1 Banco de pruebas del laboratorio de Aviónica [MIS09]. 84
II.5.2 Objetivos que se han escogido para realizar las pruebas. 85
II.5.3 Punto de inicio de las pruebas. Base Aérea de Getafe. 86
II.5.4 Circunferencia en coordenadas polares. 87
II.5.5 Secuencia de los resultados obtenidos. 89
6 Planteamiento del escenario de clasificación de una imagen. 94

III.1.1 Representación vectorial de una imagen. 96

III.2.1 Principio del algoritmo Complex-Log [Kag91]. 100
III.2.2 Ejemplo de aplicación del algoritmo CLM a imágenes rotadas. 101
III.2.3 Ejemplo de aplicación del algoritmo CLM a imágenes escaladas. 101
III.2.4 Ejemplo de aplicación del algoritmo CLM a imágenes desplazadas. 102

III.3.1 Clasificador multiclase *One vs All*. 107
III.3.2 Clasificador multiclase *Unique One vs All*. 108
III.3.3 Clasificador multiclase *All vs All*. 109

III.4.1 Banco de imágenes de la librería COIL-100. 113
III.4.2 Capturas de un mismo objeto de la base de imágenes COIL-100. 114
III.4.3 Ejemplo del procesado de una imagen. 115
III.4.4 Ejemplo de imagen con ruido gaussiano y sal y pimienta. 116

III.5.1 Arquitectura del clasificador de imágenes. 120
III.5.2 Ejemplos de espacios con módulo lineal y logarítmico. 123
III.5.3 Ejemplo de la trasformación de una imagen en los espacios con módulo lineal y logarítmico. 124
III.5.4 Ejemplo de Interpolación. 125
III.5.5 Ejemplo de kernel RBF. 126

III.6.1 Imágenes de la COIL-100 similares. 131
III.6.2 Imágenes de la COIL-100 distintas. 131
III.6.3 Evolución del % del error para imágenes distintas y similares. 133
III.6.4 Evolución de los parámetros: número de SV, C y γ. 133
III.6.5 Evolución del % error. 136
III.6.6 Evolución de los parámetros Número de SV, C y γ. 136
III.6.7 Resultados de la clasificación de imágenes con ruido. 138
III.6.8 Resultados obtenidos en los escenarios 1 y 3. 141
III.6.9 Resultados obtenidos en los escenarios 2 y 4 con saltos de 40° de rotación. . . 141
III.6.10 Resultados obtenidos en los escenarios 2 y 4 con saltos de 160° de rotación. . . 141
III.6.11 Evolución de los SV en los escenarios 1 y 3. 142
III.6.12 Evolución de los SV en los escenarios 2 y 4 con saltos de 40° y 160° de rotación. 142
III.6.13 Resultados obtenidos en el escenario 3 utilizando la fase 1 del algoritmo *Complex-Log* y el algoritmo completo. 143
III.6.14 Resultados obtenidos en los escenarios 1 y 3. 144
III.6.15 Resultados obtenidos en los escenarios 2 y 4 con saltos de 40° de rotación. . . 144
III.6.16 Resultados obtenidos en los escenarios 2 y 4 con saltos de 160° de rotación. . . 144
III.6.17 Evolución de los SV en los escenarios 1 y 3. 145

III.6.18 Evolución de los SV en los escenarios 2 y 4 con saltos de 40° y 160° de rotación. 145

IV.1.1 Ciclo de vida de un proyecto aeronáutico [Kle08]. 150
IV.1.2 Modelo en V [MIS09]. 151
IV.1.3 Módulos auxiliares [Kle08]. 153

IV.2.1 Tasa de crecimiento de las distintas cotas [HS]. 159

IV.3.1 Ejemplo planteado de integración. 172

Lista de Tablas

II.3.1 Formulación para el cálculo de las coordenadas cartesianas a partir de las coordenadas LLA. . . . 69
II.3.2 Formulación para el cálculo de las coordenadas LLA a partir de las coordenadas cartesianas. . . . 70

II.4.1 Propiedades de FSUIPC utilizadas. . . . 77

II.5.1 Pruebas del cálculo de la actitud de la cámara 82
II.5.2 Objetivos que se han escogido para realizar las pruebas. . . . 85
II.5.3 Tabla con las extrapolaciones para el FS. . . . 87
II.5.4 Pruebas del algoritmo de geo-localización. . . . 88
II.5.5 Tiempos de ejecución. . . . 90

IV.2.1 Ordenes de magnitud de la notación O. . . . 159
IV.2.2 Tiempos de ejecución de cada línea. . . . 160
IV.2.3 Tiempos de ejecución de cada línea. . . . 161
IV.2.4 Resultados del análisis de algoritmos. . . . 169

A.1 Costes directos imputables al proyecto. . . . 193
A.2 Tabla de los costes totales del proyecto. . . . 194

B.1 Lista de acrónimos (Subproyecto EADS). . . . 197
B.2 Lista de acrónimos (Subproyecto UC3M). . . . 198

PARTE I

PRELIMINARES

Capítulo I.1

Introducción

Resumen

En este capítulo, se detallan las cuestiones generales del proyecto: los motivos, el planteamiento y los objetivos del mismo. También se hará una breve explicación de los capítulos de los que consta este documento. Para finalizar, el último apartado contiene consideraciones importantes a tener en cuenta durante la lectura del documento.

La introducción va a permitir al lector/a situarse dentro del marco de desarrollo del proyecto. En primer lugar, se van a explicar los motivos por los que este proyecto se ha llevado a cabo y los objetivos que se marcaron al inicio de su desarrollo.

A continuación, se realizará un breve repaso por los puntos que componen este documento, explicando su estructura y el contenido de cada uno de los capítulos. Para finalizar, se han enumerado una serie de consideraciones, relativas a la naturaleza del proyecto y la empresa (EADS) con la que se ha colaborado para su desarrollo.

I.1.1 Planteamiento y motivación del proyecto

Desde los orígenes de la humanidad, el ser humano ha deseado volar. Han sido muchos los que han deseado volar ellos mismos o desarrollar aparatos que lo pudiesen hacer. Fueron los hermanos Wright, los que en 1903 consiguieron que volase el primer avión, el "Flyer I". Desde ese día, muchas han sido los personas que han aportado a la aeronáutica sus conocimientos y descubrimientos. Para ello, se han tomado distintos caminos, modificar la configuración, la aerodinámica, el paso de hélices a motores, utilizar distintos tipos de motores, el cambio de fuselaje...

Los mayores logros se consiguieron durante la primera y segunda guerra mundial, debido a las necesidades de los países implicados en poseer las mejores tecnologías para neutralizar al enemigo. Hoy en día, las investigaciones y los grandes avances tecnológicos han permitido llegar a desarrollar aviones no tripulados (UAVs). En la aviación comercial, los usuarios son muy reticentes a la idea de que los aviones no utilicen piloto; en cambio, en la aviación no comercial, los UAVs juegan un papel muy importante. La clave reside en las pérdidas humanas que se pueden producir si se pierde un avión pilotado. Los UAVs, al ser aviones sin piloto, en el caso de perder el avión, únicamente habría que contabilizar pérdidas materiales. Este aspecto motiva e impulsa a los desarrolladores a crear nuevos y más sofisticados UAVs.

Este proyecto surgió con un nuevo desarrollo que se está llevando a cabo en la división de *Defence & Security* de la empresa EADS, el ATLANTE. Las misiones principales que desarrollan los UAVs son de inteligencia, vigilancia, adquisición de blancos y reconocimiento. Para realizarlas, el UAV lleva incorporado un sensor EO (Electro-Óptico), que está gobernado por un computador. En él irán en un futuro implementados los algoritmos de geo-localización y geo-apuntamiento, que permitirán al operador de tierra apuntar la cámara del sensor EO a una posición específica, o localizar la posición a la que esté apuntando la cámara en un instante determinado.

Ésta es la primera fase del proyecto, desarrollada en EADS. En ella, se realizará un estudio de la transformación de las coordenadas, tanto del objetivo al cual se quiere apuntar (geo-apuntamiento) como de la posición de la cámara, que permitirá calcular el punto geográfico al que está apuntando (geo-localización). Una vez realizado el estudio, se implementarán los dos algoritmos en C++ [Str00] y se someterán a una batería de pruebas para verificar la correcta implementación de los mismos. La fase de pruebas se ha subdividido en dos fases: pruebas en PC para eliminar errores de implementación, y pruebas de integración en un rig [1] de aviónica perteneciente a EADS, para la comprobación visual de la correcta implementación de los algoritmos.

La cámara del sensor EO captura imágenes cuando el operador o algún sistema se lo indica. En el caso que se está analizando, se capturarán las imágenes a las que apunta la cámara del sensor EO, después de ejecutar los algoritmos implementados. Como segunda fase del proyecto se ha planteado el análisis de las imágenes capturadas para detectar la posible presencia o no de distintos

[1] Se denomina rig a un banco de pruebas donde puede estar reproducida parcial o totalmente la arquitectura de los computadores de un avión, en los que se integrará en el futuro las implementaciones que se desarrollan.

objetivos: fuegos, personas, aviones, edificios, convoys ...Esta segunda fase del proyecto se ha desarrollado en la UC3M. Para realizarla, se han utilizado métodos que facilitan la clasificación de imágenes. En este caso, se ha elegido un método de gran reconocimiento en cuanto a su potencial y efectividad a la hora de clasificar, llamado máquina de vectores soporte (SVM, *Support Vector Machine*).

Las imágenes por su composición y características proporcionan mucha información sobre lo que representan, de ahí el dicho "más vale una imagen que mil palabras". Debido a la cantidad de información que nos rodea, nuestro cerebro tiende a organizarla clasificando las imágenes en función de los atributos comunes. El problema que surge es que el ojo humano puede, a simple vista, perder información relevante para la clasificación.

Dentro del marco del proyecto, la imagen a la que apunta la cámara contiene mucha información para el operador de tierra. Si se la transmite a él tal y como es capturada (en crudo), las decisiones con respecto a lo que ve, serán tomadas con mucha lentitud. El procesado de las imágenes y la clasificación de los posibles objetivos contenidos en ellas constituyen una ayuda al operador de tierra en la toma de decisiones. Es importante destacar, que el sistema implementado no sustituirá en ningún momento la toma de decisiones por parte del operador, únicamente es una ayuda para él. Esto es debido a que el sistema no es un clasificador perfecto y las decisiones que se toman pueden ser de nivel crítico.

Debido a la gran carga de trabajo que tiene el operador de tierra, surgió la idea de implementar un clasificador de las imágenes capturadas para ayudarle en la toma de decisiones. El proceso consiste en capturar las imágenes, tratarlas en el UAV o en la estación de tierra, extraer la información característica de la imagen y, mediante un clasificador, determinar el tipo de objetivo que se identifica. Esta información se ha de mandar al operador de tierra, aunque también podría ser enviada a otros aviones que colaboren con el UAV o a las estaciones de tierra de otros UAVs. El objetivo principal de esta parte es implementar un clasificador de imágenes, ver la eficiencia en cuanto a la clasificación de imágenes con unas determinadas características y realizar un estudio de viabilidad de embarcar esta nueva funcionalidad en el UAV.

Para finalizar el proyecto, se ha planteado un ejercicio teórico de integración de las dos partes del proyecto como parte de un UAV, enmarcando el proyecto dentro del proceso que se realiza en un desarrollo aeronáutico real.

En la imagen I.1.1 se muestran de manera visual las dos partes en las que se divide el proyecto.

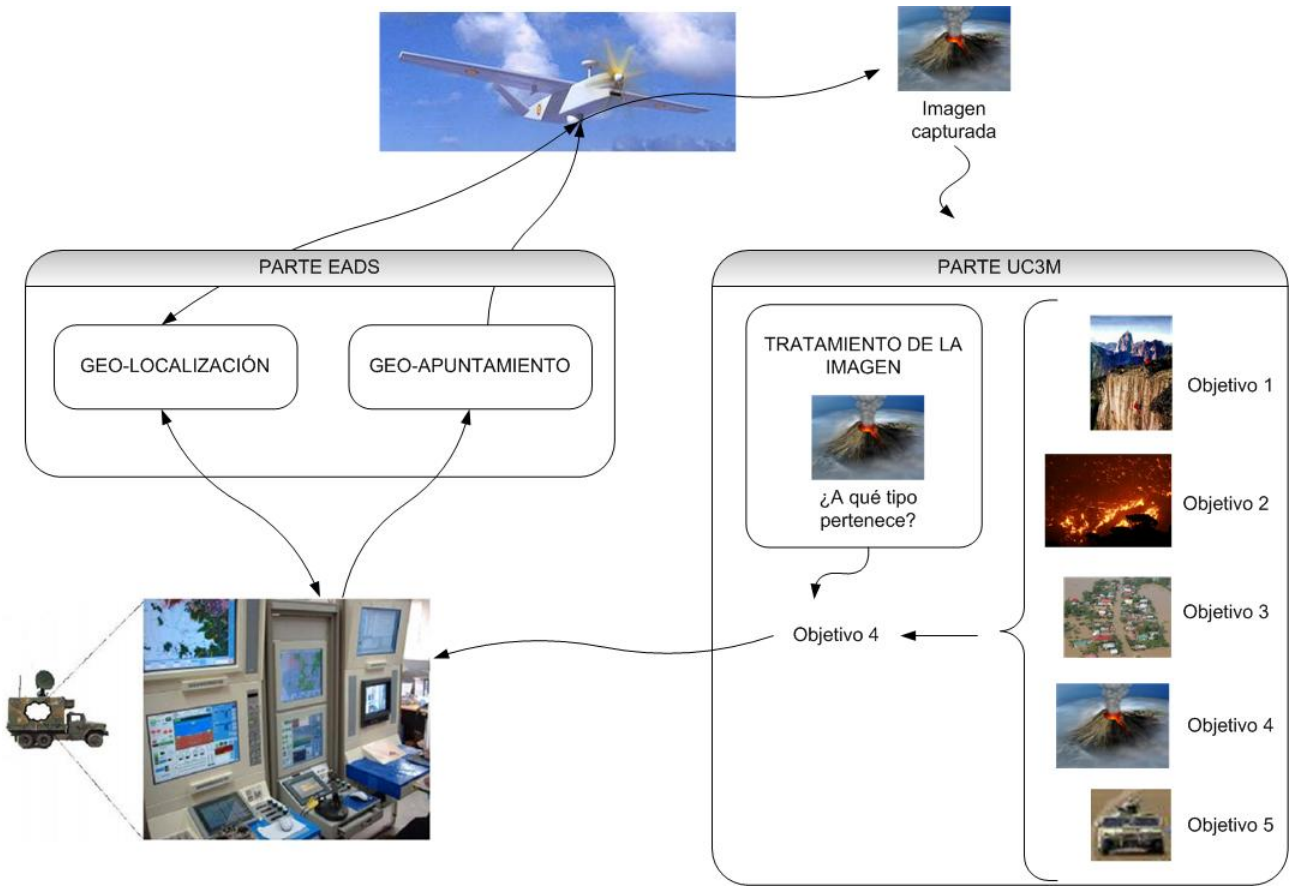


Figura I.1.1: Planteamiento inicial del proyecto.

I.1.2 Objetivos

Como se ha comentado en la sección I.1.1, el proyecto está dividido en dos subproyectos y cada parte tiene sus propios objetivos.

El objetivo principal de la primera parte es implementar y comprobar el funcionamiento de los algoritmos propuestos para el manejo de la cámara del UAV ATLANTE y, mediante una batería de pruebas, tanto en PC como de integración en un entorno de simulación, comprobar la correcta implementación y funcionamiento de los algoritmos.

El objetivo de la segunda parte es evaluar las prestaciones que nos ofrecen las SVMs en el problema de clasificación de imágenes. Debido al problema de la maldición de la dimensionalidad que poseen las SVMs, se ha determinado estudiar y aplicar un algoritmo de extracción de características (PCA) que permita reducir las dimensiones de las muestras que se utilizan, y así evitar el problema existente. Además, debido a la naturaleza de las imágenes, se planteará la utilización del algoritmo *Complex-Log* para transformar las imágenes a una representación invariante a rotaciones y escalados.

Para finalizar, se realizará un ejercicio de análisis de la posibilidad de integración de los algoritmos desarrollados en el computador que controla el sensor EO y, de cómo se podría realizar dicha integración en un UAV.

I.1.3 Visión general del documento

Tal y como se ha comentado en la sección I.1.1 y se puede ver en la figura I.1.1, el proyecto está dividido en dos subproyectos bien diferenciados. Este hecho se ve reflejado en este documento tal y como se describe a continuación:

Parte 1. La primera parte permite al lector situarse en el entorno en el que se va a desarrollar el proyecto. En esta parte, se realiza una introducción al proyecto explicando los objetivos que se quieren alcanzar, la motivación que han llevado a realizarlo y las partes que componen el documento que se está escribiendo. Además se explican los antecedentes que existían con anterioridad al proyecto y que son necesarios para su realización: qué es el UAS (*Unmanned Aerial System*), los UAVs, las SVMs y nociones básicas de cartografía y tratamiento de imágenes.

- **Capítulo 1 - Introducción**.
- **Capítulo 2 - Antecedentes**.

Parte 2. La segunda parte se centra en el desarrollo del subproyecto realizado en EADS. Este subproyecto consiste en el análisis e implementación de los algoritmos de geo-apuntamiento y geo-localización, que en un futuro irán implementados en el computador que comande la cámara que lleva instalado el UAV. A lo largo de los capítulos que componen esta parte, se explican los fundamentos teóricos que se han utilizado, las implementaciones y las pruebas que se han realizado.

- **Capítulo 1 - Sistemas de referencia**.
- **Capítulo 2 - Algoritmos para el control de sensores EO**.
- **Capítulo 3 - Proyección cónica conforme de Lambert**.
- **Capítulo 4 - El módulo FSUIPC (*Flight Simulator Universal Inter-Process Communication*)**.
- **Capítulo 5 - Plan de pruebas**.

Parte 3. Esta parte analiza el segundo subproyecto, en el que se va a implementar un clasificador de imágenes y a evaluar la posibilidad o no de embarcarlo en el avión. Antes de clasificar la imagen, es necesario procesarla, aplicar el algoritmo PCA para reducir la dimensionalidad del problema y el algoritmo *Complex-Log* para convertir las imágenes a una representación invariante a rotaciones. A lo largo de los capítulos que componen esta parte, se explican las técnicas utilizadas, la implementación realizada y los resultados obtenidos.

- **Capítulo 1 - Extracción de características (PCA)**.
- **Capítulo 2 - Algoritmo *Complex-Log Mapping* (CLM)**.
- **Capítulo 3 - Clasificadores multiclase**.
- **Capítulo 4 - Imágenes empleadas (COIL-100)**.
- **Capítulo 5 - Clasificadores de imágenes mediante SVM**.
- **Capítulo 6 - Análisis de resultados**.

Parte 4. En esta parte, se va a realizar un ejercicio de integración del desarrollo, dentro del marco de un proyecto aeronáutico en la división de *Defence & Security* de EADS. En el primer capítulo, se va a definir el ciclo de vida de un proyecto, la metodología de trabajo y la normativa a tener en cuenta durante los desarrollos. El segundo y tercer capítulo se centran en una de las fases más importantes del proyecto, la integración. Primero, se realiza un estudio de los recursos computacionales que requieren los algoritmos. Asi se verifica la posibilidad de integración de los algoritmos en los computadores del UAV, esta fase se denomina integración a nivel de componente *software*. A continuación, se plantea un esquema de integración de los algoritmos dentro del computador que controla el sensor EO y los sistemas con los que va a interactuar, esta fase se denomina integración a nivel de sistemas.

Capítulo 1 - Ciclo de vida de un proyecto de *software* aeronáutico.

Capítulo 2 - Integración a nivel de componente *software*.

Capítulo 3 - Integración a nivel de sistemas.

Parte 5. En esta parte, se exponen las conclusiones y los trabajos futuros de este proyecto. Las conclusiones son el capítulo más importante del proyecto, en él se hace balance del trabajo realizado, analizando los objetivos propuestos al inicio del proyecto y comprobando si han sido alcanzados o no. En los trabajos futuros se plantearán las líneas de trabajo a seguir a partir del trabajo realizado en este proyecto.

Capítulo 1 - Conclusiones.

Capítulo 2 - Trabajos futuros.

Parte 6. Para finalizar el documento se han añadido dos apéndices: un presupuesto del coste que podría suponer para una empresa realizar este proyecto y una lista de los acrónimos utilizados durante el desarrollo del documento.

Apéndice 1 - Presupuesto.

Apéndice 2 - Acrónimos.

I.1.4 Consideraciones del proyecto

Este proyecto se ha desarrollado en colaboración con la empresa EADS, en la división de *Defence & Security*. Debido a la actividad que se desarrolla en la empresa y, en concreto en esta división, hay consideraciones importantes a tener en cuenta durante la realización de este proyecto.

- Por razones de propiedad intelectual, confidencialidad y seguridad, no está permitido incluir el código desarrollado para el control de sensores EO en UAVs.
- Debido al entorno en el que se ha desarrollado el proyecto, muchos términos se denominan por siglas, ya que los nombres que lo designan son muy largos. Para facilitar la lectura del documento, se ha desarrollado la lista B.1, en el apéndice B, donde se especifican los acrónimos asociados a cada sigla.

Capítulo I.2

Antedecentes

Resumen

En este capítulo, se va a realizar un recorrido por los conocimientos previos para la realización del proyecto.
En la primera parte del capítulo, se explicará qué es un UAS, cómo funciona y de qué está compuesto. Una de las partes más importantes del UAS es el UAV, se explicará qué es, las características que posee y se mostrarán ejemplos de UAVs. Se ha escogido el UAV ATLANTE como UAV de referencia para este proyecto. Esto permitirá al lector/a, según avancen los capítulos, entender qué aporta este proyecto al avión y por qué es beneficioso para él. Además para el desarrollo de los algoritmos, es necesario tener nociones básicas de cartografía; éstas, también serán explicadas en este capítulo.
En la segunda parte del capítulo, se explicará qué es el aprendizaje máquina, los tipos que existen, cuáles son las ventajas e inconvenientes. Esto permitirá entender el funcionamiento de las SVMs que se explicarán a continuación.

Es fundamental conocer el entorno en el que se va a desarrollar el proyecto. En este caso, se está desarrollando para los aviones no tripulados (UAVs), de los cuales se ha tomado como ejemplo el UAV ATLANTE. Es importante conocer el sistema donde desarrollan sus funciones los UAVs (UAS), qué son, sus características más importantes y los ambientes en los que desarrollan sus misiones, para así adecuar lo más posible el proyecto a ellos.

Para el desarrollo de la primera parte del proyecto, es necesario tener nociones básicas de cartografía, para después comprender en su totalidad los algoritmos de geo-apuntamiento y geo-localización que se desean desarrollar.

Como parte final del proyecto, se desea implementar un clasificador para las imágenes proporcionadas por el sensor EO del UAV; para ello, se ha optado por utilizar las SVMs, que son máquinas de aprendizaje.

I.2.1 Sistemas aéreos no tripulados (UAS)

La filosofía en la que se basan los UAVs es la ausencia de piloto y tripulación a bordo de los mismos. Este hecho conlleva la desaparición de algunos sistemas del avión como los asientos, paneles de instrumentación, sistemas de acondicionamiento de aire, sistemas de oxígeno... y la sustitución de las actuaciones directas sobre la plataforma, por comandos desde la estación de tierra. Por todo ello surge la necesidad de una infraestructura que compense estas ausencias, el UAS (en inglés, *Unmanned Aerial System*) [dDE09]. Fue el departamento de Defensa de los EEUU el encargado de introducir este término, que más tarde, fue adoptado por la FAA (*Federal Aviation Administration*).

El UAS es un sistema formado por un segmento de vuelo y otro de tierra. El segmento de vuelo está compuesto por la plataforma aérea, la carga útil según la misión a desarrollar y el sistema de comunicaciones. Por su parte, el segmento de tierra está compuesto por la estación de tierra; desde la cual, es posible controlar la aeronave y su carga de pago, comunicarse con la aeronave y analizar la información de los sensores de la plataforma aérea. Además, la plataforma debe poder ser lanzada y recuperada con seguridad e integridad para volver a ser utilizada. A continuación se explican brevemente cada uno de los componentes de un UAS:

- **La plataforma aérea** es el vehículo no tripulado (UAV) (ver figura I.2.1). Tiene tamaños muy variables dependiendo de la misión que ha de desarrollar. Posee diferentes sistemas de sustentación (ala fija, rotatorias...) y diferentes sistemas de propulsión (turbohélices, turborreactores, motores eléctricos...). Además, incorpora los sistemas de propulsión, navegación, comunicaciones y los *data links*. Éstos, permiten realizar el control del vuelo, el control de la misión y transmitir la información de los sensores desde el UAV a la estación de tierra. Se desarrollarán en detalle en la sección I.2.2.

Figura I.2.1: Ejemplo de plataforma aérea, UAV ATLANTE [EAD10].

- **La carga útil o *payload*** está formada por los equipos embarcados en la plataforma aérea

que no son esenciales para el vuelo de la misma. Por ejemplo, el sensor EO para el que se está desarrollando este proyecto (ver figura I.2.2)

Figura I.2.2: Ejemplo de carga útil, sensor EO [Sen10].

- **La estación de control de tierra (GCS, *Ground Control Station*)** es junto con el UAV una de las partes más importantes del sistema. Son cabinas transportables que alojan en su interior los equipos de comunicaciones, procesado de datos, cálculo, visualización, monitorización y control de los datos que manda el UAV. Además, desde la estación base se puede gestionar, controlar y comandar el UAV. En ella se desarrollan los planes de misión, se controlan los sensores embarcados, se modifican los planes de vuelo, se comunica con el UAV para la transmisión de información y alguna plataforma es posible controlarla remotamente. En la figura I.2.3 se puede ver un ejemplo de estación base.

Figura I.2.3: Ejemplo de estación de tierra.

- **El sistema de lanzamiento y recuperación** (LRS, *Launch and Recovery System*) es el encargado de controlar el UAV durante las fases de taxi, despegue, fase inicial del vuelo, aproximación y aterrizaje. El sistema de lanzamiento es una plataforma que lanza el vehículo mediante diferentes técnicas : catapulta, neumática, hidráulica, cohete... En la figura I.2.4 se muestran algunos ejemplos.

Figura I.2.4: Ejemplos de sistemas de lanzamiento y recuperación.

En la figura I.2.5 se puede ver un ejemplo de escenario de operación de un UAV. Para que pueda salir a volar es necesario realizar los siguientes pasos:

- El operador de la estación de tierra genera el plan de vuelo que ha de llevar a cabo el UAV. En este plan de vuelo se especifica la misión que ha de desarrollar el UAV y se carga mediante comunicaciones seguras desde la estación de control de tierra. Además del plan de vuelo, la estación ha de especificarle al UAV la carga de pago que posee y algunos parámetros propios de la misión. Por ejemplo, para este proyecto, se debería cargar con el plan de misión, las matrices de características y vectores soporte que contiene la información para clasificar objetivos.
- El UAV sale a volar, para ello se utiliza el subsistema de lanzamiento.
- En la mayoría de las ocasiones el UAV puede realizar el vuelo de dos modos, vuelo por control remoto desde la estación base o vuelo automático (ver sección I.2.2.2). En el vuelo por control remoto, es el operador de la estación de tierra el que vuela el avión mediante los controles de la estación. En el vuelo automático, el UAV está gobernado por sus sistemas de control y no es necesaria la intervención del operador de la estación de tierra. Desde la estación base se puede modificar el plan de misión en cualquier momento, pilotar, manejar los sensores...
- Durante todo el tiempo que dura la misión, el UAV obtiene información de sus sensores y de los satélites con los que se comunica. Toda esta información ha de ser transmitida a la estación de tierra para que allí sea procesada.
- Una vez llevada a cabo la misión, el UAV retorna a la base y aterriza. Aquí entra en operación el subsistema de recuperación.

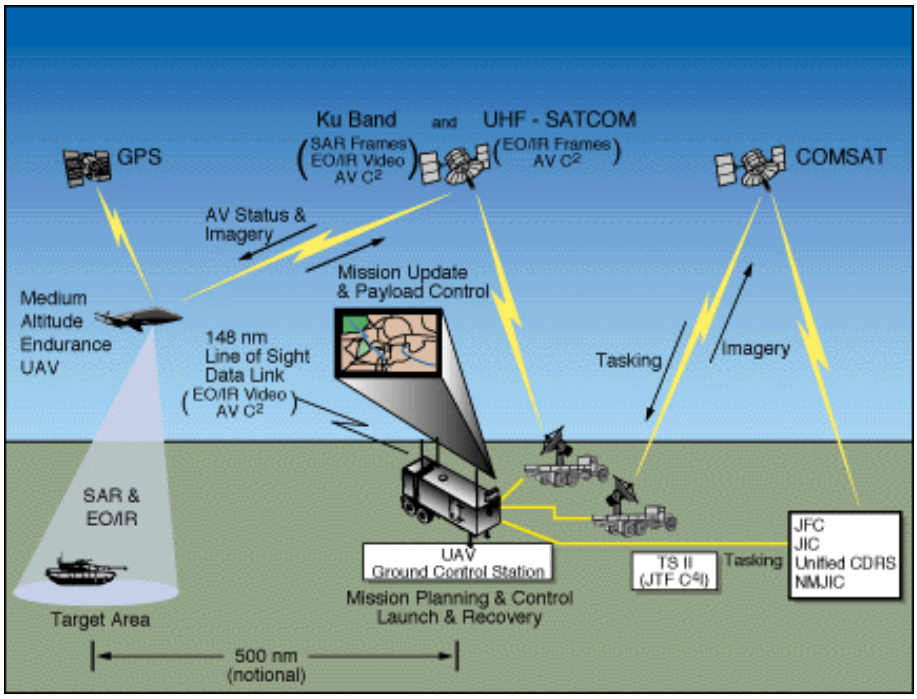


Figura I.2.5: Ejemplo de escenario de un sistema UAS.

I.2.2 Los aviones no tripulados (UAVs)

Se denominan UAVs (*Unmanned Aerial Vehicles*) o VANTs (Vehículos Aéreos No Tripulados), a los vehículos que son capaces de volar sin la necesidad de un piloto humano a bordo, gracias a un sistema de pilotaje autónomo o por control remoto desde la estación de tierra. El abanico de dispositivos aéreos que se pueden clasificar dentro de esta categoría, es muy amplio; desde los cometas, hasta aviones controlados por radio, pasando por misiles.

El primer ejemplar de UAV se desarrolló después de la primera guerra mundial. Este ejemplar sirvió para entrenar a los operarios de cañones antiaéreos. Es en 1985, cuando EEUU lanza el primer UAV con completa autonomía; y son los militares de los EEUU los primeros que denominaron a este tipo de aviones UAVs. Es importante resaltar que EEUU es el país que más ejemplares posee.

En España, el desarrollo de los UAVs comenzó un poco más tarde, en 1988, cuando el INTA (Instituto Nacional de Técnica Aeroespacial) pensó en desarrollar un sistema de vigilancia del campo de batalla nacional. El proyecto comenzó a desarrollarse con vistas a posibles conflictos bélicos. Para ello se basaron en el principio:

> *"Ninguna actuación sobre territorio enemigo podrá tener éxito sin que previamente se haya asegurado el espacio aéreo sobre él. Es necesario el dominio del espacio aéreo, bien para negar el poder aéreo enemigo, para poder lanzar ataques tácticos o estratégicos o para poder observar los movimientos del enemigo."*

Teniendo en la mente esta premisa, el INTA comenzó la investigación y el desarrollo de los UAVs, que culminó en el año 2006, cuando se entregaron 4 ejemplares al ejército español. Actualmente, hay aproximadamente unos 6700 UAVs operando en la OTAN (Organización del Tratado del Atlántico Norte).

Los UAVs se pueden utilizar en muchos y diversos tipos de escenarios; por ejemplo: en ambientes de alta toxicidad química y radiológica (el desastre de Chernóbil, donde la toma de muestras y las tareas de control del ambiente, conllevaban un alto grado de peligrosidad para las vidas humanas), en misiones de control del narcotráfico y contra el terrorismo, grabación de vídeos de alta calidad para ser empleados como pruebas... Para poder realizar estas misiones, es necesario que cumplan las normas establecidas en el Tratado de Cielos Abiertos de 1992, que permiten los vuelos de UAVs sobre todo el espacio aéreo de sus signatarios.

Actualmente, las empresas aeronáuticas están centrando sus investigaciones y desarrollos en los UAVs. Se tiende a que todo esté controlado por el computador de vuelo y que la carga de los pilotos sea cada vez menor hasta llegar a desaparecer. En la aviación no comercial, su implantación es fundamental para evitar la pérdida de vidas humanas. Mientras tanto, en la aviación comercial, existe más reticencia a implantar aviones sin piloto, aunque todas las tendencias apuntan a que en unos años será una realidad.

I.2.2.1 Características y misiones que desarrollan

Una de las aplicaciones más extendidas de los UAVs es la de realizar misiones de reconocimiento y vigilancia. Hasta este momento, estas misiones eran realizadas por aviones caza, ya que poseen potentes cámaras capaces de detectar los movimientos de los enemigos; pero presentaban algunos inconvenientes que el uso de UAVs ha permitido resolver:

- Posibilidad de pérdidas humanas si el avión es detectado y por tanto derribado.
- Agotamiento de los pilotos por la larga duración de las misiones.
- Imposibilidad de permanencia durante un largo periodo sobre territorio enemigo sin riesgo de ser detectados.

Una vez vistas las ventajas que nos ofrecen este tipo de aviones, se van a enumerar los tipos de misiones en las que pueden tomar parte, tanto en el entorno militar como en el entorno civil.

- **Entorno militar**
  - Vigilancia de posiciones enemigas, costas y fronteras.
  - Reconocimiento y localización precisa de objetivos.
  - Control de fuego propio sobre el enemigo (corrección de línea de tiro).
  - Repostaje en vuelo.
  - Combates aéreos.
  - Escolta aérea.
  - Guerra electrónica.
  - Comunicaciones (como relays para internet o telefonía movil)
- **Entorno civil**
  - Detección y control de incendios.
  - Control de tráfico de carreteras y comunicaciones.
  - Rescate de náufragos.
  - Control de bancos de pesca, cosechas, entorno ecológico y narcotráfico.
  - Situaciones de emergencia y catástrofes.
  - Reconocimiento del terreno.
  - Meteorología y Oceanografía.

Después de muchos estudios y desarrollos, se han conseguido importantes mejoras en los UAVs para el desarrollo de misiones, las cuales se enumeran a continuación:

1. Permanencia de más de 7 horas en el aire.
2. Transporte de distintos tipos de carga útil en función de la misión que ha de realizar: sensores de visión nocturna, sensores infrarrojos, radares de apertura sintética en desarrollo...
3. Posibilidad de obtener, manejar y transmitir información gracias a la aplicación de técnicas de protección (guerra electrónica, criptografía...); lo que posibilita realizar comunicaciones más seguras, difíciles de detectar e interferir.

I.2.2.2 Clasificación

Existen muchas y diversas maneras de clasificar los UAVs. A continuación se muestras las clasificaciones más utilizadas:

1. **Atendiendo a su misión principal.**
   - *De blanco*, sirven para simular aviones o ataques enemigos en los sistemas de defensa de tierra o aire.
   - *De reconocimiento (URAVs, Unmanned Reconnaissance Aerial Vehicles)*, envían información militar.
   - *De combate (UCAVs, Unmanned Combat Aerial Vehicles)*, participan en conflictos bélicos y llevan a cabo misiones que suelen ser muy peligrosas.
   - *Tácticos (UTAVs, Unmanned Tactical Aerial Vehicles)*, realizan misiones tácticas.
   - *Logística*, diseñados para llevar carga.
   - *Investigación y desarrollo (Demostrator)*, en ellos se prueban e investigan los sistemas en desarrollo.
   - *UAVs comerciales y civiles*, son diseñados para propósitos civiles.
2. **Atendiendo al origen de la misión.**
   - *Civil*
   - *Militar*
3. **Atendiendo al peso.**
   - *Super pesados*, más de 2000 Kg.
   - *Pesados*, entre 200 y 2000 Kg.
   - *Medianos*, entre 50 y 200 Kg.
   - *Ligeros*, entre 5 y 50 Kg.
   - *Micros*, inferior a 5 Kg.
4. **Atendiendo a la cota de vuelo o altitud**
   - *Muy alta cota (VHA, Very Hight Altitude)*
   - *Alta cota (HA, Hight Altitude)*, viaja por encima de los 45.000 pies [1] de altitud.
   - *Media cota (MA, Medium altitude)*, viaja entre 15.000 y 45.000 pies de altitud.
   - *Baja cota (LA, Low Altitude)*, viaja por debajo de los 15.000 pies de altitud.
5. **Atendiendo a la duración de la misión**
   - *Larga duración (LE, Long Endurance)*, realizan misiones de 24 horas o más.
   - *Media duración (ME, Medium Endurance)*, realizan misiones entre 5 y 24 horas.
   - *Corta duración (SE, Short Endurance)*, realizan misiones de menos de 5 horas.
6. **Atendiendo al tipo de control**

[1] 1 pie = 0.3048 metros

- *Autónomo y adaptativo o automático*, el UAV está totalmente gobernado por sus sistemas de abordo, sin intervención del operador de tierra. El UAV tiene la capacidad de re-planificar su plan de misión en función de los cambios producidos durante el desarrollo de la misma. El UAV puede interactuar con otros UAVs.
- *Monitorizado*, el UAV opera de forma autónoma. El operador controla la retroalimentación del UAV; pero no controla los mandos, aunque puede tomar decisiones.
- *Supervisado*, el UAV realiza algunas operaciones de forma autónoma. El control recae en su gran mayoría sobre el operador de tierra.
- *Autónomo-no adaptativo (o pre-programado)*, el UAV obedece a una rutina programada con anterioridad, no tiene capacidad de cambiarla para adaptarla al desarrollo de la misión.
- *Mando directo por un operador o control remoto*, el UAV responde directamente a los mandos del operador de tierra.

7. **Mezcla de las anteriores**

- *HALE (High Altitude, Long Endurance)*, viaja por encima de los 30.000 pies de altitud y alcance indeterminado.
  - CIS Lunar, viaja entre la luna y la tierra.
  - ORBITAL, viaja en órbitas bajas terrestres (Mach [2] 25+)
  - *HYPERSONIC* (supersónico (Mach 1-5) o hipersónico (Mach 5+)), viaja a 50.000 pies de altitud o altitud suborbital y tiene un alcance de 200 km aprox.
- *MALE (Medium Altitude, Long Endurance)*, viaja entre 20.000 y 30.000 pies de altitud y tiene un alcance de unos 200 km aprox.
- *TACTICAL (Low Altitude, Short Endurance)*, viaja por debajo de los 18.000 pies de altitud y tiene un alcance por debajo de 160 km aprox.
  - NATO: viaja a 10.000 pies de altitud y hasta 50 km de alcance aprox.
  - *Closed*, viaja a 5.000 pies de altitud y 10 km de alcance aprox.
  - *Handheld*, viaja a 2.000 pies de altitud y 2 km de alcance aprox.

I.2.2.3 Los UAVS de EADS

El proyecto, como se ha comentado en la introducción, se ha desarrollado en colaboración con la empresa EADS, empresa europea de referencia para el sector aeronáutico. En esta sección, se va a dar una pincelada de los UAVs que se han desarrollado o que están en desarrollo en la empresa a día de hoy.

Como se ha comentado, es uno de los negocios más importantes del sector aeronáutico en EADS, y en concreto de la división de *Defence & Security*. La tabla I.2.6 muestra los UAVS desarrollados, en desarrollo o en proyecto de desarrollo de esta división.

[2] Número adimensional usado para describir la velocidad de los aviones. Mach 1 equivale a la velocidad del sonido, subsónico ($M < 0,7$), transónico ($0,7 < M < 1,2$), Supersónico ($1,2 < M < 5$) e Hipersónico ($M > 5$)

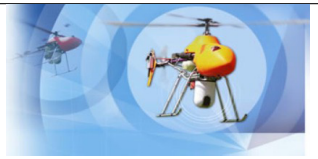
 *EuroHawk* EuroHawk GmbH [EADS y Northrop] 2010	 *Agile UAV-NCE* Within Network-Centric Environments 2013	 *Barracuda (UAV Demonstrator)* UCAV 2006
 *Programa SIDM* SIDM (Système Intérimaire de Drone MALE) 2008	 *DRAC/tracker* Drone de Renseignement Au Contact 2010	 *Scorpio (Rotary-wing UAV Demonstrator)* UAV de ala giratoria y una torreta giroestabilizada
 *Sharc (Rotary-wing UAV Demonstrator)* EADS y Vertivision - VTOL (Vertical Take-Off and Landing) 2008/09	 *Orka (Rotary-wing UAV Demonstrator)* Drone de Renseignement Au Contact 2006	 *Neuron (UCAV Demonstrator)* Dassault En desarrollo
 *CL-289* URAV Años 90	 *Advanced* MALE UAV En desarrollo	 **Atlante** MALE UTAV En desarrollo

Figura I.2.6: Proyectos de UAVs en los que participa EADS [[EAD10] y [Cor09]].

I.2.2.4 El ATLANTE (Avión Táctico de Largo Alcance No Tripulado Español)

El ATLANTE [[EAD10] y [Cor09]] se ha escogido como avión de referencia entre los UAVs desarrollados en EADS para la realización de este proyecto. Es llamado el UAV Español y es un avión táctico de largo alcance (UTAV MALE). El diseño se está llevando a cabo en la división de *Defence & Security* y será utilizado por el Ejército de Tierra español, para llevar a cabo misiones ISTAR (*Intelligence, Surveillance, Target Acquisition and Reconnaissance*).

Atendiendo a las distintas clasificaciones explicadas en el apartado I.2.2.2, el ATLANTE se clasifica como un **MALE UTAV** que realiza misiones **ISTAR** y vuela en modo automático.

Otras operaciones que el ATLANTE puede llevar a cabo son: la identificación de blancos, vigilancia diurna y nocturna, reconocimiento sobre colinas, evaluación de daños en una zona de batalla, protección de tropas y convoys, vigilancia de fronteras, control de tráfico de drogas y búsqueda y rescate. El UAV también contará con algunos requisitos del departamento de seguridad nacional, la guardia civil y otras agencias de emergencia española.

El ATLANTE puede operar en distintos escenarios, que puede estar formado por 4 o más vehículos aéreos, una estación de control terrestre, una terminal de datos terrestre, una unidad de transporte, lanzamiento y recogida, una terminal de vídeo remoto y una unidad de mantenimiento. Además, podrá ofrecer a los operadores de tierra, información en tiempo real del campo de batalla del enemigo, para la realización de la vigilancia y adquisición de blancos en grandes superficies.

I.2.2.4.1 Diseño del ATLANTE

El ATLANTE tendrá la capacidad de operar 24 horas al día, bajo cualquier condición climática. No necesita pistas de aterrizaje/despegue, aunque esté provisto de tren de aterrizaje, el cual está pensado para poder operar en pistas improvisadas. Está diseñado para funcionar en dos modos, el aterrizaje y despegue en pistas o despegue con un iniciador y recuperación con un paracaídas. El avión podrá ser controlado desde la estación de tierra o a través de un modo automático.

El Atlante estará equipado con sensores EO (Electro-Ópticos) e IR (Infrarrojos). El sensor EO convertirá los rayos de luz en señales electrónicas para la captura de imágenes en tiempo real y vídeos. Este es el sensor para el que se está desarrollando este proyecto.

Las características más importantes son:

- Dimensiones
  - Longitud: 4.60 m
  - Altura: 1.26 m
  - Envergadura: 8m
- Pesos
  - Máxima carga útil: 100kg
  - Máximo peso al despegue: 520kg
  - Envergadura: 8m
- Comportamiento
  - Altura de servicio: 20.000 pies

- Rango: 160Km
- Máxima resistencia: 20h

En la figura I.2.7 se muestran algunas imágenes del prototipo del avión.

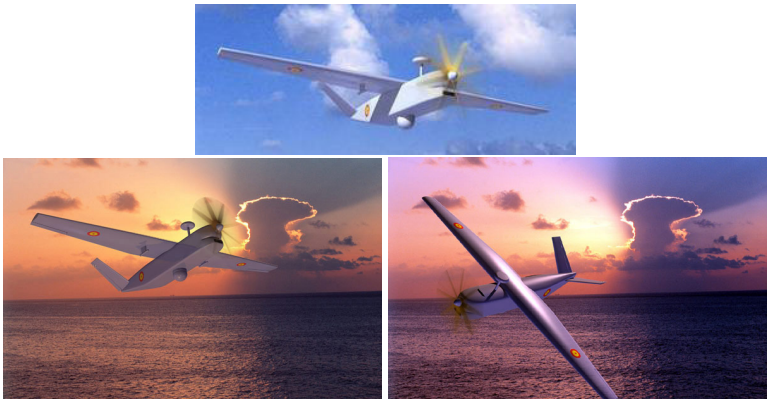


Figura I.2.7: Prototipos del UAV Atlante [[EAD10] y [Cor09]].

I.2.3 Nociones básicas de cartografía

Según la RAE (Real Academia Española), la cartografía se define como el arte de trazar mapas geográficos y la ciencia que los estudia.

La representación de la información de la superficie terrestre en cualquier soporte plano (por ejemplo en un mapa) es fundamental que se haga de manera correcta. Hasta ahora se ha realizado siempre por geógrafos, cartógrafos y expertos del sector, para que no se produzcan propagaciones de errores en el manejo de la información; ya que, un mínimo error a la hora de la representación supone un error muy importante (durante las pruebas realizadas se ha podido comprobar que una variación de centésimas supone una representación errónea en el simulador, ver II.5.2). Existe bastante confusión en la utilización de algunos términos básicos en cartografía, lo que puede inferir errores de comprensión; por este motivo, en esta sección se van a unificar y clarificar los conceptos necesarios para este proyecto [[GWP01] y [Reu]].

Coordenada ⇒ Cantidad lineal o angular que designa la posición que un punto ocupa en un sistema de referencia. También se usa como término general para designar un tipo particular de sistema de referencia, tales como las coordenadas cartesianas o coordenadas esféricas.

Coordenadas geográficas ⇒ El sistema de coordenadas geográficas es un sistema de referencia que utiliza dos coordenadas angulares, la latitud y la longitud, para designar la posición de un punto sobre la superficie terrestre o en cualquier superficie esférica.

La representación de unas coordenadas de la Torre Eiffel en París son: 48° 51’29.95” N, 2° 17’40.18” E. En este proyecto todos los cálculos se realizarán con las coordenadas en decimal. En este caso serían: 48.858319°, 2.294494°

En la figura I.2.8 se puede ver como un punto de la superficie terrestre queda designado por sus coordenadas geográficas, la latitud y la longitud.

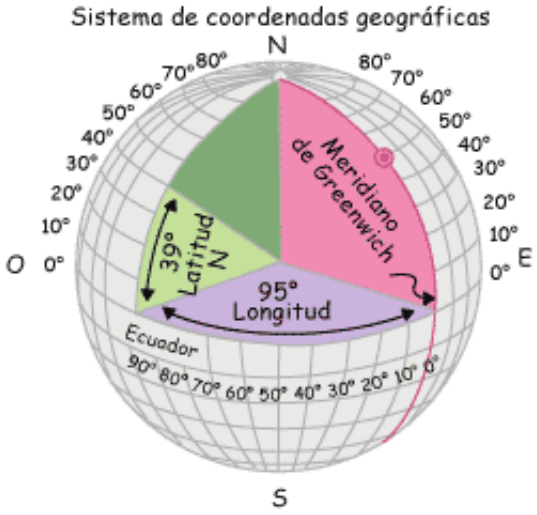


Figura I.2.8: Ejemplo de representación de un punto P sobre la superficie terrestre mediante la latitud y la longitud.

Coordenadas geográficas geocéntricas ⇒ Coordenadas cartesianas (X, Y y Z) que representan la posición de un punto con respecto al centro de masas de la tierra.

Coordenadas geodéticas $\Rightarrow$ Valor de latitud, longitud y altura (φ, λ y h) que definen la posición de un punto en la superficie de la tierra con respecto al elipsoide de referencia.

Altura geodética, elipsoidal (h) $\Rightarrow$ Altura sobre el elipsoide de referencia, medida a lo largo de la normal elipsoidal a través del punto en cuestión. La altura geodética es positiva si el punto está fuera del elipsoide.

Longitud geodética (λ) $\Rightarrow$ Ángulo entre el plano de un meridiano y el del meridiano cero. Una longitud puede ser medida desde el ángulo formado entre los meridianos local y cero, al polo de rotación del elipsoide de referencia, o por el arco a lo largo del Ecuador interceptado por esos meridianos. La longitud es 0° en el meridiano de referencia, meridiano cero o meridiano de Greenwich, positiva hacia el este del meridiano de Greenwich, negativa hacia el oeste del mismo, siendo los valores extremos +180° y -180° (ver figura I.2.8)

Latitud (φ) (ver figura I.2.8)

- **Latitud geocéntrica** (φ_{GEOC})**.** Es el ángulo entre el plano ecuatorial y una línea desde el punto de referencia al centro de la tierra. Hasta la llegada del GPS (*Global Positioning System*), este ángulo no era posible medirlo con precisión. Esta latitud no va a ser utilizada en el proyecto.
- **Latitud geodética** (φ_{GEOD})**.** Es el ángulo entre el plano ecuatorial y la normal al elipsoide a través del punto medido. La latitud es 0° en el ecuador, positiva en el hemisferio norte y negativa en el hemisferio sur. Siendo los valores de los Polos Norte y Sur, +90° y -90° respectivamente.
- **Latitud paramétrica** (φ_{PARAM})**.** No tiene sentido físico. Se utiliza como soporte matemático para las transformaciones entre las latitudes geocéntrica y geodética.

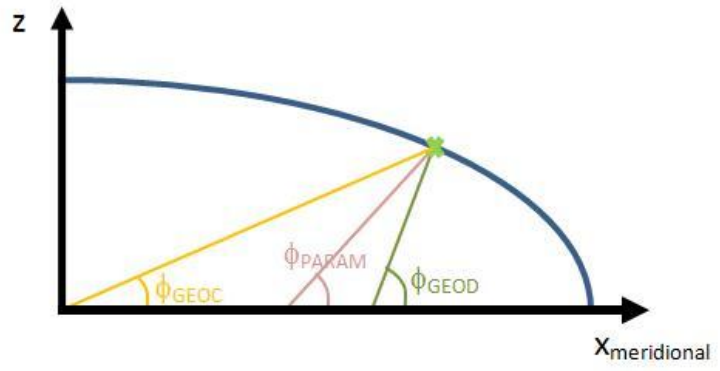


Figura I.2.9: Latitudes geocéntrica, paramétrica y geodética en el plano meridional.

Meridiano $\Rightarrow$ Línea de referencia de norte a sur, sobre cada uno de los círculos máximos de la esfera terrestre que pasan por los polos, desde donde se determinan las longitudes y azimuts; o las intersecciones con los planos que forman un gran círculo que contiene ambos polos geográficos de la tierra y el elipsoide.

Meridiano cero o meridiano de Greenwich $\Rightarrow$ Meridiano desde el que se calculan todas las longitudes de todos los meridianos. Este meridiano tiene longitud 0°, y se elige por pasar a través del observatorio de Greenwich (Inglaterra). En una nueva redefinición del sistema de coordenadas, la localización del meridiano cero se estableció por el Servicio Internacional de rotación de la tierra en París (Francia).

Paralelo ⇒ Línea sobre la tierra o una representación de ella, que representa la misma latitud en cada punto.

Ecuador o paralelo cero ⇒ Línea de latitud geodética cero; círculo máximo descrito por el semieje mayor del elipsoide de referencia, como si rotase sobre el semieje menor.

Elipsoide ⇒ Superficie generada por un elipsoide rotando sobre uno de sus ejes. También llamada elipsoide de revolución.

Elipsoide de referencia ⇒ Elipsoide cuyas dimensiones cerradas se aproximan a las de un geoide; las dimensiones exactas son determinadas por varias consideraciones concernientes a la sección de la superficie de la tierra. Normalmente es un elipsoide de referencia bi-axial.

I.2.3.1 Cálculo de la latitud paramétrica (φ_{PARAM})

La latitud paramétrica no tiene sentido físico, únicamente es útil para relacionar las latitudes geocéntricas y geodéticas, las cuales si tienen sentido físico. Está basada en elipsoides de revolución, donde cada meridiano es una elipse con un radio ecuatorial o semieje mayor a y un radio polar o semieje menor b.

La figura I.2.10 representa los tres tipos de latitudes y las relaciones trigonométricas existentes, en ellas está basada la formulación que se presenta a continuación.

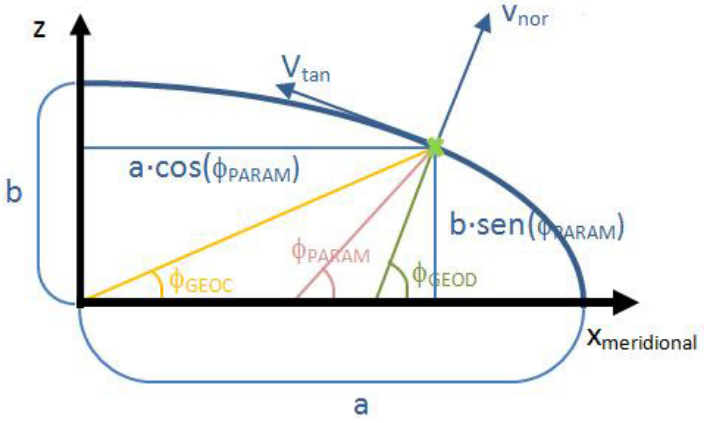


Figura I.2.10: Latitudes geocéntrica, paramétrica y geodética en el plano meridional.

$$\frac{x^2_{MERIDIONAL}}{a^2} + \frac{z^2}{b^2} = 1 \tag{I.2.1}$$

$$x_{MERIDIONAL} = a\cos(\varphi_{PARAM}) \tag{I.2.2}$$

$$z = a\,\mathrm{sen}(\varphi_{PARAM}) \tag{I.2.3}$$

$$\cos^2(\varphi_{PARAM}) + \mathrm{sen}^2(\varphi_{PARAM}) = 1 \tag{I.2.4}$$

Sustituyendo entre ellas las ecuaciones I.2.1, I.2.2, I.2.3 y I.2.4 obtenemos:

$$\frac{a^2\cos^2(\varphi_{PARAM})}{a^2} + \frac{b^2\sin^2(\varphi_{PARAM})}{b^2} = 1 \tag{I.2.5}$$

I.2.3.2 Cálculo de la latitud geodética (φ_{GEOD})

Se define la latitud geodética como el ángulo por encima o por debajo del plano ecuatorial con la normal a la superficie de la elipse en el punto. Es posible establecerla en función de φ_{PARAM} porque es ortogonal a la dirección meridional tangencial. El vector tangencial al meridiano $\overrightarrow{v_{TAN}}$, estará en la dirección de la derivada de la solución de la ecuación de la elipse con respecto a la latitud paramétrica, y la dirección meridional normal $\overrightarrow{v_{NOR}}$, será ortogonal a ésta. Viendo la figura I.2.10 se deducen,

$$\overrightarrow{v_{TAN}} \alpha \frac{\partial}{\partial \varphi_{PARAM}} \begin{pmatrix} a\cos(\varphi_{PARAM}) \\ b\sin(\varphi_{PARAM}) \end{pmatrix} = \begin{pmatrix} -a\sin(\varphi_{PARAM}) \\ b\cos(\varphi_{PARAM}) \end{pmatrix} \tag{I.2.6}$$

$$\overrightarrow{v_{NOR}} \perp \overrightarrow{v_{TAN}} = \begin{pmatrix} b\cos(\varphi_{PARAM}) \\ a\sin(\varphi_{PARAM}) \end{pmatrix} \tag{I.2.7}$$

La tangente de φ_{GEOD} es el cociente entre la componente z y la componente x del vector de superficie normal:

$$\tan\varphi_{GEOD} = \frac{z}{x} = \frac{a\sin(\varphi_{PARAM})}{b\cos(\varphi_{PARAM})} = \frac{a}{b}\tan(\varphi_{PARAM}) \tag{I.2.8}$$

Utilizando relaciones trigonométricas, se obtiene:

$$\sin(\varphi_{GEOD}) = \frac{\tan(\varphi_{GEOD})}{\sqrt{1+\tan^2(\varphi_{GEOD})}} = \frac{\frac{a\sin(\varphi_{PARAM})}{b\cos(\varphi_{PARAM})}}{\sqrt{1+\frac{a^2\sin^2(\varphi_{PARAM})}{b^2\cos^2(\varphi_{PARAM})}}} = \frac{\frac{a\sin(\varphi_{PARAM})}{b\cos(\varphi_{PARAM})}}{\sqrt{\frac{b^2\cos^2(\varphi_{PARAM})+a^2\sin(\varphi_{PARAM})}{b^2\cos^2(\varphi_{PARAM})}}} =$$

$$\frac{a\sin(\varphi_{PARAM})}{\sqrt{b^2\cos^2(\varphi_{PARAM})^2 + a^2\sin(\varphi_{PARAM})}} \tag{I.2.9}$$

Siguiendo el mismo procedimiento, se obtiene:

$$\cos(\varphi_{GEOD}) = \frac{1}{\sqrt{1+\tan^2(\varphi_{GEOD})}} = \frac{b\cos(\varphi_{PARAM})}{\sqrt{a^2\sin^2(\varphi_{PARAM}) + b^2\cos(\varphi_{PARAM})}} \tag{I.2.10}$$

$$\sin(\varphi_{PARAM}) = \frac{\tan(\varphi_{PARAM})}{\sqrt{1+\tan^2(\varphi_{PARAM})}} = \frac{b\sin(\varphi_{GEOD})}{\sqrt{a^2\cos^2(\varphi_{GEOD}) + b^2\sin(\varphi_{GEOD})}} \tag{I.2.11}$$

$$\cos(\varphi_{PARAM}) = \frac{1}{\sqrt{1+\tan^2(\varphi_{PARAM})}} = \frac{a\cos(\varphi_{GEOD})}{\sqrt{a^2\cos^2(\varphi_{GEOD}) + b^2\sin(\varphi_{GEOD})}} \tag{I.2.12}$$

Por tanto, utilizando las ecuaciones I.2.2, I.2.3, I.2.11 y I.2.12 se obtiene:

$$x_{MERIDIONAL} = a\cos(\varphi_{PARAM}) = \frac{a^2\cos(\varphi_{GEOD})}{\sqrt{a^2\cos^2(\varphi_{GEOD}) + b^2\sin(\varphi_{GEOD})}} \tag{I.2.13}$$

$$z = b\sin(\varphi_{PARAM}) = \frac{b^2\sin(\varphi_{GEOD})}{\sqrt{a^2\cos^2(\varphi_{GEOD}) + b^2\sin(\varphi_{GEOD})}} \tag{I.2.14}$$

Estas ecuaciones se aplican únicamente en la superficie del geoide. Para aplicarlas a distintas alturas, es necesario utilizar la altura ortométrica h, la cual está definida por encima o por debajo del geoide y es medida a lo largo de la superficie normal. De tal manera, las coordenadas $x_{MERIDIONAL}$ y z de un punto con altura h son (ver I.2.11):

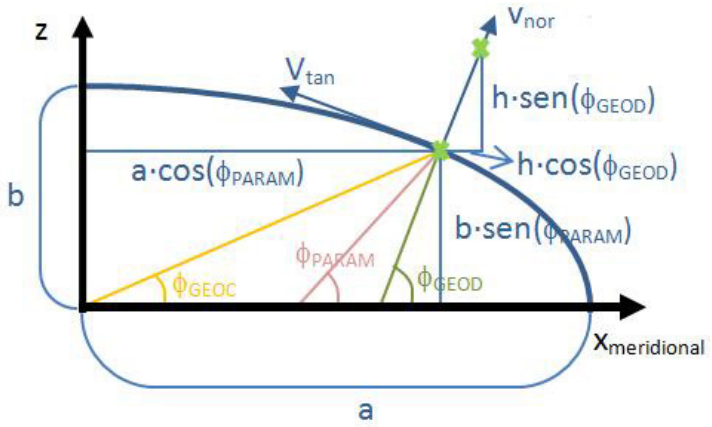


Figura I.2.11: Latitudes geocéntrica, paramétrica y geodética de un punto de altura h.

$$x_{MERIDIONAL} = a\cos\varphi_{PARAM} + h\cos(\varphi_{GEOD}) = a\frac{a\cos(\varphi_{GEOD})}{\sqrt{a^2\cos^2(\varphi_{GEOD}) + b^2\sin(\varphi_{GEOD})}} + h\cos(\varphi_{GEOD}) =$$

$$\cos(\varphi_{GEOD})\left(h + a^2\frac{1}{\sqrt{a^2\cos^2(\varphi_{GEOD}) + b^2\sin^2(\varphi_{GEOD})}}\right) \qquad \text{(I.2.15)}$$

$$z = b\sin(\varphi_{PARAM}) + h\sin(\varphi_{GEOD}) = h\sin(\varphi_{GEOD}) + b\frac{b\sin(\varphi_{GEOD})}{\sqrt{a^2\cos^2(\varphi_{GEOD}) + b^2\sin(\varphi_{GEOD})}} =$$

$$\sin(\varphi_{GEOD})\left(h + b^2\frac{1}{\sqrt{a^2\cos^2(\varphi_{GEOD}) + b^2\sin^2(\varphi_{GEOD})}}\right) \qquad \text{(I.2.16)}$$

A partir de este momento se consideran siempre las latitudes, **latitudes geodéticas**, excepto que se especifique explícitamente.

I.2.4 Teoría del tratamiento digital de información

Hasta hace unas décadas, pensar que un computador podría aprender, era inconcebible. La RAE define aprendizaje como "la acción y efecto de aprender algún arte, oficio u otra cosa", el "tiempo que en ello se emplea"y "adquisición por la práctica de una conducta duradera".

Herbert Simon afirma que "Aprender produce cambios en un sistema, y son estos cambios los que permiten al sistema hacer la misma tarea de manera más eficiente la siguiente vez."

Toda la teoría de tratamiento estadístico de información se puede agrupar en dos conjuntos,

Problemas de decisión: en estos problemas, el objetivo es decidir a qué clase pertenece una muestra entre las posibles clases disponibles. En el ámbito del aprendizaje máquina, consiste en encontrar la regla de decisión durante el entrenamiento de la máquina, que permita clasificar una nueva muestra correctamente dentro del conjunto de clases posibles.

Problemas de regresión: en este tipo de problemas, el objetivo es estimar el valor de alguna variable desconocida, pero cuyo patrón es el mismo que el conjunto de observaciones conocidas a priori.

Los problemas de decisión, y en concreto el aprendizaje máquina [Mit97], comprende diferentes mecanismos, reglas, enfoques y tecnologías; mediante los cuales, un ordenador puede aprender a desarrollar tareas que los seres humanos hacen de forma natural y rápida, por ejemplo: reconocer imágenes, tomar decisiones...

El principio del aprendizaje máquina, consiste en introducir una serie de ejemplos con los que ella aprende; y durante su uso, cuando reciba nuevos datos, poder clasificarlos de manera correcta según lo aprendido anteriormente.

El primer problema que se plantea, es la necesidad de un conjunto de muestras de entrenamiento. El conjunto debe ser representativo del problema que se desea tratar, y esto no siempre es posible. Este proyecto es un claro ejemplo, ante la imposibilidad de adquirir un conjunto de muestras reales, los experimentos han tenido que realizarse con una base de datos de imágenes, y se han tratado para que posean las características de las imágenes reales. Los *outliers*, las variables irrelevantes y las redundantes, pueden alterar el comportamiento correcto de la máquina, creando indecisiones o incertidumbre en el diseño; por ello, es necesario tenerlas bajo control.

La cantidad de muestras, tal y como se ha dicho, ha de ser representativa. Hay situaciones en las que es necesario un amplio número de muestras, debido a la criticidad del problema, y se necesita que la clasificación se realice de manera muy precisa. En otras ocasiones, el hecho de tener muchas muestras, no hace más que aumentar la carga computacional y disminuir la precisión de la clasificación. Estas decisiones pueden llevarnos a problemas de sobreajuste y subajuste, que se deben evitar, para que el funcionamiento de la máquina sea correcto.

Existen dos tipos de aprendizaje máquina, supervisado y no supervisado [Hay99]. En ambos, se tiene conocimiento de las muestras de entrada de entrenamiento; pero mientras en el supervisado se conocen también las etiquetas asociadas a las muestras, en el no supervisado no se requiere conocerlas, ya que el sistema en sí, es el que se organiza para identificar los grupos de datos mediante las características comunes entre ellos. Las SVMs implementan un tipo de aprendizaje máquina supervisado, por lo que se va a hacer más hincapié en este tipo de aprendizaje.

Aprendizaje no supervisado: En el aprendizaje no supervisado, se suministran a la máquina únicamente las muestras de entrenamiento, pero sin las etiquetas donde se especifica el patrón al que pertenece cada una.

Este aprendizaje es utilizado principalmente para descubrir rasgos significativos en el conjunto de muestras, extraer rasgos o agrupar patrones (*clustering*) mediante la estimación de la densidad de probabilidad de los datos,...

Aprendizaje supervisado: En el aprendizaje supervisado, se conocen las etiquetas asociadas a las muestras de entrenamiento; es decir, las salidas deseadas. El diseñador de la máquina, ha de introducir las entradas y comparar el resultado obtenido con la salida deseada; generalmente, el resultado no coincidirá y se generará un error de salida.

En este proyecto la máquina que se pretende entrenar va a utilizar un aprendizaje supervisado, para ello se han de seguir los siguientes pasos:

Paso 1 - Entrenamiento. Esta es la fase importante de la máquina, en ella se introduce un conjunto de muestras representativo con las correspondientes etiquetas, con ellas la máquina es entrenada. En esta fase, la máquina aprende las relaciones existentes entre las entradas y las salidas del problema a partir de los ejemplos proporcionados.

Paso 2 - Validación. Del conjunto de muestras de entrenamiento, un subconjunto se utiliza para, mediante validación cruzada, establecer los parámetros óptimos de la máquina. Esta fase no siempre se considera separada de la anterior, en ocasiones se integra con la fase de entrenamiento, tal y como ocurre en este proyecto.

Paso 3 - Test. Se introducen una serie de muestras, no vistas en la fase de entrenamiento, y se comparan los resultados obtenidos con los esperados. Así se evalúa como es de buena la máquina y la capacidad de generalización que posee.

Durante mucho tiempo, el hombre ha querido replicar la estructura del cerebro humano, debido a que éste realiza rutinariamente tareas de reconocimiento de patrones. Desde pequeños, los humanos realizamos un aprendizaje continuo, aprendemos a reconocer objetos y caras de la gente cercana. El cerebro humano es capaz de discernir en una imagen, una cara no familiar, en pocos milisegundos. A través de la experiencia, el cerebro crea sus propias reglas para clasificar las imágenes que recibe.

Muchos han sido los científicos que se han dedicado estudiar las redes neuronales; pero la auténtica revolución, fue el algoritmo de retropropagación propuesto inicialmente por Werbos y más tarde popularizado gracias a Rumelhart en 1986. Este algoritmo permite el entrenamiento de perceptrones multicapa sin realimentación.

I.2.5 Máquinas de vectores soporte (SVMs)

En 1963 Vapnik y Lerner [VL63] propusieron una clase computacionalmente poderosa de redes de aprendizaje supervisado, llamadas Máquinas de Vectores Soporte (SVMs). Las SVMs permiten resolver problemas de reconocimiento de patrones, regresión y estimación de densidades; pero no fue hasta los años 90 cuando se desarrolló y se generalizó.

En esta sección se van a explicar los conceptos básicos de las SVMs. Si se desea profundizar más sobre ellas, las siguientes referencias son las que se han utilizado como base para el estudio y comprensión de las mismas las referencias: [SBS98], [PV07], [Sch00], [HCL09], [svm10] y [SS02]

Las SVMs permiten producir un modelo, con el cual es posible predecir el tipo de instancia de cada dato del conjunto de test. Dados un conjunto de puntos pertenecientes a dos clases (caso binario), la SVM es capaz de encontrar el hiperplano que deja la mayor parte de puntos de una misma clase en el mismo lado, a la vez que maximiza la distancia entre el punto más cercano de cada clase y el hiperplano, y mantiene una buena capacidad de generalización de la máquina.

Para conseguir una buena generalización hay que evitar los efectos de sobreajuste y subajuste, tipicos de los entrenamientos máquina. Estos dos efectos se ven representados en la figura I.2.12.

El sobreajuste se produce cuando el modelo está muy ajustado a los datos de entrenamiento y ante nuevas muestras no responde correctamente. El modelo no es capaz de generalizar y se suele dar con fronteras muy complejas. Este efecto se puede ver en la imagen superior derecha, en la figura I.2.12.

El subajuste se produce cuando la frontera de decisión escogida es muy sencilla, debido a la inexistencia de muestras representativas en el conjunto de entrenamiento; y por tanto los resultados que se obtienen ante nuevas muestras, no son muy buenos. Se puede apreciar en la imagen superior izquierda de la figura I.2.12.

Un entrenamiento óptimo es el representado en la parte inferior en la figura I.2.12, ya que logra una mejor clasificación de las nuevas muestras y aumenta la capacidad de generalización de la máquina con respecto al resto de posibilidades.

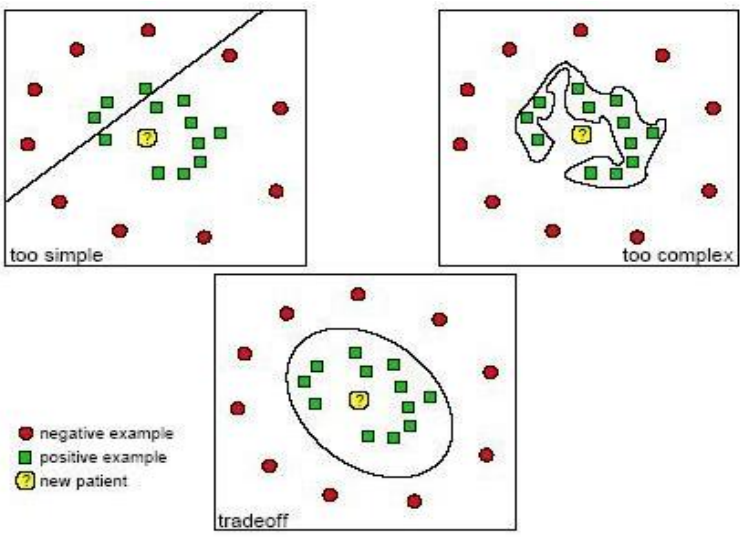


Figura I.2.12: Ejemplos de los efectos del sobreajuste y subajustes [svm10].

El objetivo de la SVM es encontrar el hiperplano de separación óptima, llamado OSH (*Optimal Separating Hyperplane*), el cual minimiza el riesgo de clasificación errónea; pero no sólo de los datos de entrenamiento sino también de los de test. Una de las ventajas de las SVMs es la buena capacidad de generalización, incluso cuando el número de datos es muy bajo, y el espacio de entrada es de dimensiones altas.

En la imagen inferior de la figura I.2.12 se puede ver un ejemplo del hiperplano para un problema linealmente no separable. Por su parte, en la figura I.2.13 se pueden ver, para un problema linealmente separable, distintos ejemplos de hiperplanos.

Debido al concepto de maximización del margen (se explicará más adelante), las SVM siempre escogerán como hiperplano óptimo el representado en color morado. Con este hiperplano la posibilidad de clasificar correctamente una nueva muestra es mucho mayor que con el resto de hiperplanos.

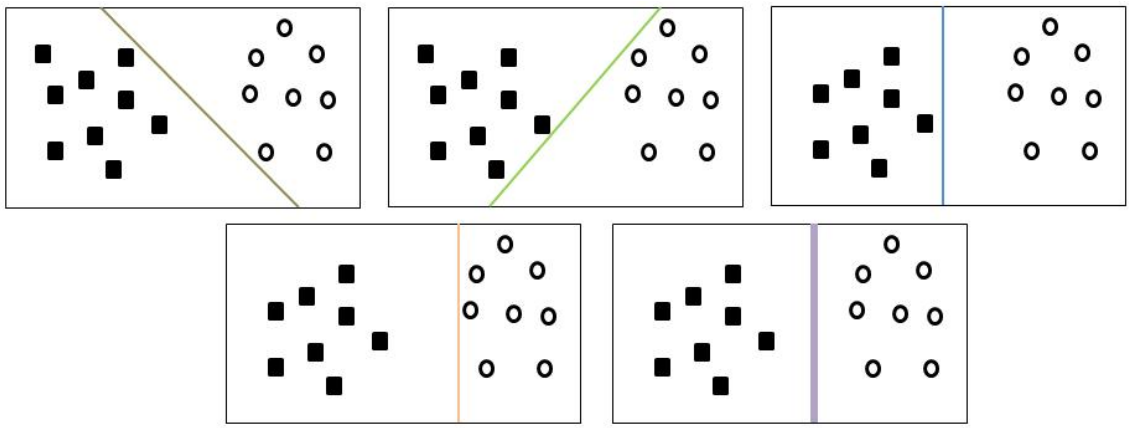


Figura I.2.13: Ejemplos de diferentes hiperplanos.

I.2.5.1 Caso linealmente separable

Esta parte de la sección se va a centrar en estudiar el caso linealmente separable. Para mayor simplicidad se va a considerar el caso binario. Partiendo de un conjunto M de muestras ($x_i \in R^n$, siendo $i = 1, 2, 3, \ldots, N$) y las etiquetas asociadas a cada una de ellas ($y_i \in +1, -1$, siendo $i = 1, 2, 3, \ldots, N$), se quiere calcular la expresión del hiperplano de separación que divide al conjunto de datos M, dejando las muestras que pertenecen a la misma clase, al mismo lado del hiperplano, y maximizando la distancia entre cada clase y el hiperplano calculado (también llamado margen, ver figura I.2.14). N es el número de muestras que existen en el conjunto de datos M.

Un conjunto es linealmente separable si existe un valor $w \in R^n$ que cumple

$$y_i(w^T x_i + b) \geq 1, \text{ siendo } i = 1, 2, 3, \ldots, N. \tag{I.2.17}$$

Esta ecuación se puede dividir en las siguientes dos ecuaciones:

$$(w^T x_i + b) > 1, \text{ si } y_i = 1 \tag{I.2.18}$$
$$(w^T x_i + b) < -1, \text{ si } y_i = -1 \tag{I.2.19}$$

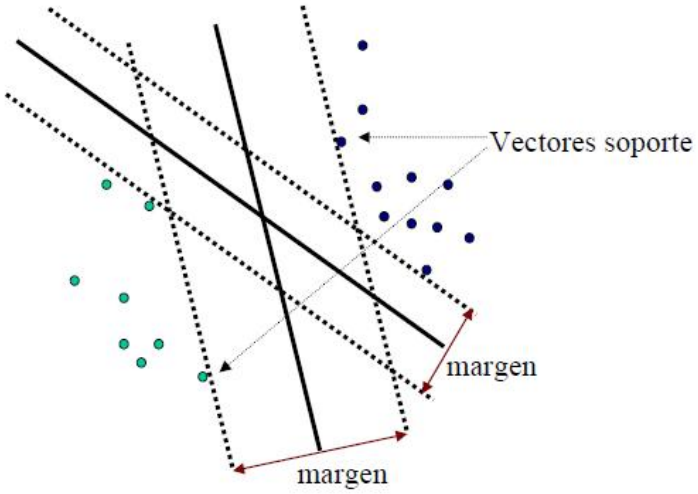


Figura I.2.14: Ejemplo de maximización del margen mediante la elección del OSH.

Un clasificador es lineal si su función de decisión puede expresarse como una función lineal de x, lo cual implica que la ecuación del hiperplano de separación cumple:

$$w^T x + b = 0 \quad w \in R^n, b \in R \tag{I.2.20}$$

donde,

- w es perpendicular al hiperplano
- x son los datos que se desean clasificar
- b es una constante llamada sesgo o *bias*
- $\frac{b}{||w||}$ es la distancia perpendicular desde el hiperplano al origen.

El clasificador buscado queda representado por la siguiente expresión:

$$f(x) = sign(w^T x) + b \tag{I.2.21}$$

Esta función indica cómo se realiza la clasificación de los datos x dependiendo del signo que tome la función al evaluarla.

En el diseño de los clasificadores máquina, es muy importante buscar que la máquina sea capaz de generalizar ante un problema de clasificación de muestras nuevas. Para aumentar esta capacidad, lo que se intenta es maximizar la distancia entre los datos y la frontera de decisión. Esta distancia queda definida por:

$$d_i = \frac{w^T x_i + b}{||w||} \tag{I.2.22}$$

aplicando la ecuación I.2.17 a la ecuación I.2.22 obtenemos que:

$$y_i d_i \geq \frac{1}{||w||} \tag{I.2.23}$$

donde $\frac{1}{||w||}$ es el límite inferior de la distancia entre los puntos de una clase y el hiperplano de separación.

Debido a la condición de un conjunto linealmente separable, el hiperplano óptimo de separación, es aquel en el que la distancia a los puntos más cercanos del conjunto M es máxima. Esta distancia queda determinada por la cota inferior calculada en la ecuación I.2.23. Esto implica que para calcular el OSH basta con minimizar la siguiente expresión, siempre sujeto a las restricciones de la ecuación I.2.20:

$$\tau(w) = \frac{1}{2}||w||^2 \tag{I.2.24}$$

Para resolver el problema anteriormente planteado, se va a utilizar el método de los Multiplicadores de Lagrange; siendo los valores de los multiplicadores de Lagrange $\alpha = (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_N) > 0$. Esto permite representar el hiperplano deseado como combinación lineal de los datos, son las llamadas coordenadas duales. Teniendo en cuenta las restricciones de la ecuación I.2.20, la expresión a minimizar que se obtiene es:

$$L = \frac{1}{2}||w||^2 - \sum_{i=1}^{N} \alpha_i[y_i(w_T x_i + b) - 1] \tag{I.2.25}$$

La solución óptima del problema, pasa por minimizar la ecuación I.2.25; lo que implica derivar la expresión, también llamada función de coste, con respecto a cada una de las variables, w y b, e igualar las expresiones obtenidas a cero.

$$\frac{\eth L}{\eth b} = \sum_{i=1}^{N} y_i \alpha_i = 0 \tag{I.2.26}$$

$$\frac{\eth L}{\eth w} = w - \sum_{i=1}^{N} y_i \alpha_i x_i = 0 \longrightarrow w = \sum_{i=1}^{N} y_i \alpha_i x_i \tag{I.2.27}$$

La SVM elige los N vectores soporte (SV, *Support Vectors*), que son aquellos que cumplen que α_i sea distinto de cero. Esta condición se cumple para las muestras más cercanas al hiperplano óptimo y al final las que definen una clase después de entrenar la máquina.

Si se sustituyen las ecuaciones I.2.26 y I.2.27 en la ecuación I.2.21, se obtiene una nueva función de decisión del hiperplano,

$$f(x) = sign\left(\sum_{i=1}^{N} y_i \alpha_i x_i^T x + b\right) \tag{I.2.28}$$

La imagen I.2.15 muestra un ejemplo de clasificación binario. En ella se muestran en rojo los vectores soporte, y en verde el hiperplano de separación entre clases óptimo. Se puede ver también en ella, que el hiperplano óptimo (en verde) es ortogonal a la línea más corta que conecta los puntos convexos de las dos clases (línea punteada), e intersecciona en la mitad entre las dos clases.

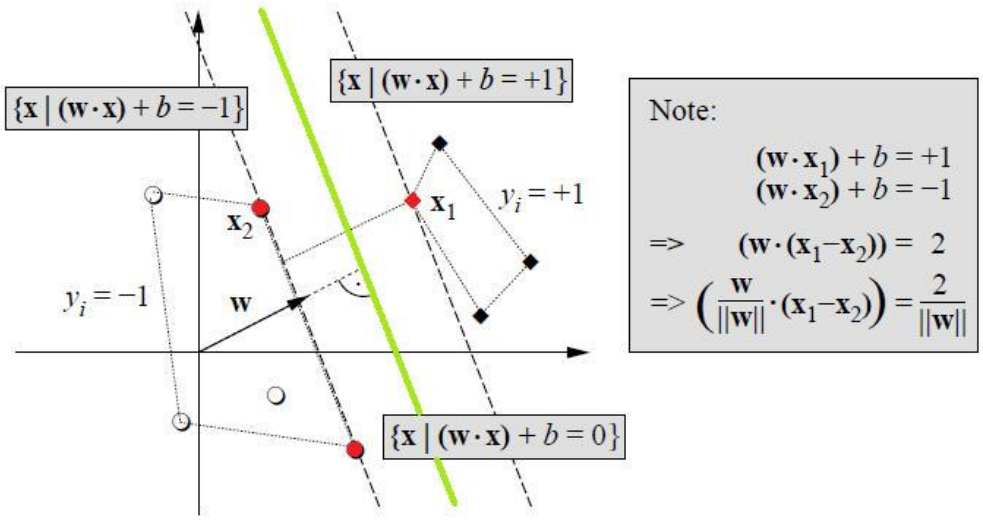


Figura I.2.15: Ejemplo de clasificación binaria [SBS98].

I.2.5.2 Caso no linealmente separable

Ahora se va a plantear el caso más general, en el que los datos no son linealmente separables. Esto implica que los datos de ambas clases (considerando el caso binario) están mezclados y por tanto el hiperplano de separación no puede tener forma lineal. Este efecto, puede surgir por la incursión de ruido o distorsiones en las muestras. El desarrollo se va a realizar de forma análoga al realizado en la sección I.2.5.1.

Para solucionar este problema, se introducen las variables $\xi_i > 0 \forall i$, también llamadas holguras, que minimizan el número de muestras mal clasificadas. Estas variables ayudan a controlar el error permitido mediante la introducción de una penalización a las muestras mal clasificadas. Esto da lugar a las siguientes modificaciones en la ecuación I.2.17:

$$y_i(w^T x_i + b) \geq 1 - \xi_i \tag{I.2.29}$$

Esta ecuación se puede descomponer en las siguientes ecuaciones:

$$(w^T x_i + b) \geq 1 - \xi_i \ \ para \ \ y_i = 1 \tag{I.2.30}$$
$$(w^T x_i + b) \leq -1 + \xi_i \ \ para \ \ y_i = -1 \tag{I.2.31}$$

En la figura I.2.16 se pueden ver dos muestras dentro del margen, pero con holguras distintas.

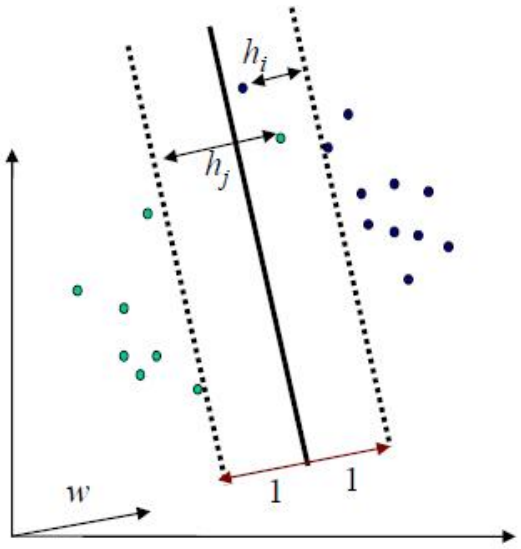


Figura I.2.16: Ejemplo de clasificación no linealmente separable.

En el dibujo la holgura h_i, se encuentra dentro del margen de su clase, entre el OSH y la recta paralela al OSH que pasa por la muestra más cercana ($0 < h_i < 1$); por lo que la muestra asociada se encuentra dentro de los límites y por tanto se da como bien clasificada.

Por el contrario la holgura h_j se encuentra fuera de los márgenes de su clase ($h_j > 1$), por lo que la muestra no está correctamente clasificada y será necesario aplicarle un factor correctivo.

La función a optimizar, modificada, queda de la siguiente manera:

$$\tau(w, \xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} \xi_i \tag{I.2.32}$$

siempre sujeto a la ecuación I.2.29.

El propósito del término extra $C\sum_{i=1}^{l} \xi_i$, es mantener bajo control el número de muestras mal clasificadas. Este término da robustez a la solución, ya que hace que OSH sea menos sensible a los *outliers* del conjunto de entrenamiento. El parámetro C o constante de penalización, permitirá llegar a un compromiso entre la complejidad del problema y el número de datos no separables. El OSH tenderá a maximizar la mínima distancia $\frac{1}{w}$ para valores pequeños de C, y a minimizar el número de muestras mal clasificadas para valores grandes de C.

I.2.5.3 SVM no lineales

Lo ideal sería poder resolver todos los casos como si fueran linealmente separables, pero cuando no lo son surgen los problemas. Para facilitar la separación de los datos, se puede recurrir a una técnica, cuya idea principal consiste en mapear los datos de entrada linealmente no separables, a un espacio de producto vectorial de dimensiones mayores pero en el que el problema sea linealmente separable. El espacio F obtenido se denomina *Feature Space* (espacio de características o espacio de medida) y la transformación que se aplica para obtener el nuevo espacio F se designa como $\varphi(x)$. Este efecto puede verse en la imagen I.2.17

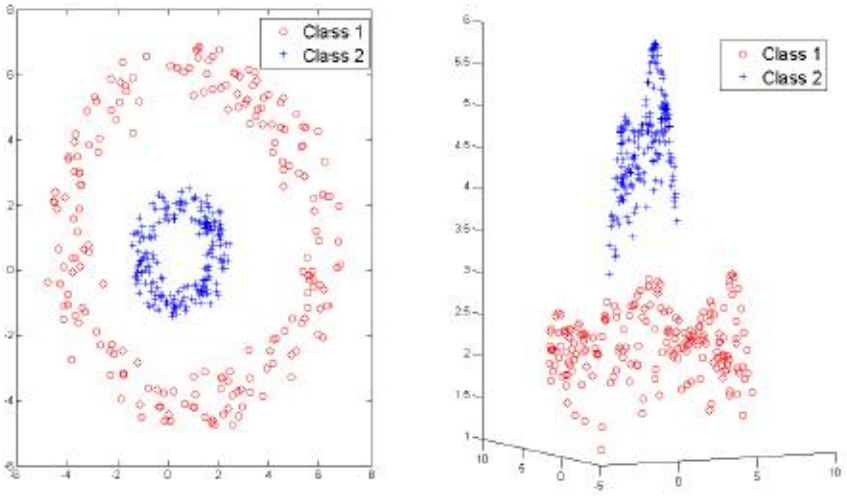


Figura I.2.17: Transformación del espacio de entrada.

Básicamente, consiste en realizar la transformación de $x \rightarrow \varphi(x)$ para todas las muestras del espacio de entrada. Por lo que la formulación es exactamente la misma que en el caso linealmente separable, pero cambiando x por $\varphi(x)$. Hay que tener en cuenta que si $\varphi(x)$ es de dimensiones muy altas, la transformación es muy costosa; y sería ideal no tener que realizar la transformación para cada muestra del espacio.

$$f(x) = \text{sign}\left(\sum_{i=0}^{N} \alpha_i y_i (\varphi(x) \cdot \varphi(x_i)) + b\right) \tag{I.2.33}$$

El operador $\cdot$ es el operador productor escalar, también conocido como producto interno, interior o punto. Es una operación externa definida sobre un espacio vectorial cuyo resultado al operar entre sí dos vectores, es un escalar o número. El producto escalar de dos vectores en un espacio euclídeo se define como el producto de sus módulos por el coseno del ángulo que forman.

La solución a la transformación depende del producto escalar que se realiza en la trasformación. Si se cumple el teorema de Mercer, el producto escalar puede escribirse como:

$$k(x_1, x_2) = (\varphi(x_1) \cdot \varphi(x_2)) \tag{I.2.34}$$

La definición de la función núcleo se hace detalladamente en la subsección I.2.5.3.1.

La condición de Mercer indica cuando una función puede ser utilizada como función núcleo y dice así:

> Existe un mapa $\varphi(x)$ y una expansión $k(x_1, x_2) = (\varphi(x_1) \cdot \varphi(x_2))$ sí y sólo sí, para cualquier $g(x)$ se cumple:
> si $\int g(x)^2 dx$ es finita, entonces, $\int k(x, y) g(x) g(y) dx dy \geq 0$, semidefinida positiva.

Debido a la dificultad de verificar el teorema para cualquier g, si éste se verifica para cualquier g con norma L_2 finita, se acepta como demostración para cualquier g.

Si se sustituye la ecuación I.2.34 en la ecuación I.2.33 se obtiene:

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i k(x, x_i) + b\right) \tag{I.2.35}$$

siendo:

- N es el número de vectores soporte.
- x son las muestras que se desean evaluar.
- x_i son los vectores soportes obtenidos durante el entrenamiento de la máquina (subconjunto de las muestras de entrenamiento)
- α_i son los multiplicadores de Lagrange obtenidos durante la fase de entrenamiento.
- y_i son las etiquetas de las muestras de entrenamiento.
- $\alpha_i y_i = \beta_i$, que se obtiene durante la fase de entrenamiento.
- b parámetro de algunas versiones de la SVM que se obtiene durante la fase de entrenamiento. En la librería LIBSVM no se utiliza.

I.2.5.3.1 La función núcleo $k(x, x_i)$

La función $k(x, x_i)$ es la denominada función núcleo o *kernel*. Ésta permite a la SVM extender la clase de funciones de decisión al caso no lineal. Schölkopf estableció que la función núcleo definía una medida de distancia en el espacio de entrada.

Existen diferentes funciones núcleo, y entre las más utilizadas se encuentran:

- Función núcleo lineal: $k(x, x_i) = x \cdot x_i$
- Función núcleo sigmoidal: $k(x, x_i) = \tanh(\gamma x \cdot x_i + c)$
- Función núcleo polinomial: $k(x, x_i) = (\gamma x \cdot x_i + c)^{\alpha}$
- Función núcleo gaussiana de base radial (RBF, *Radial Basis Function*): $k(x, x_i) = exp(-\gamma(|x \cdot x_i|)^2)$

En todas ellas γ es la constante de proporcionalidad, cuyo rango de valores útiles debe ser estimado para cada aplicación en particular, c es un coeficiente y α el grado del polinomio.

Varios resultados han determinado que la función núcleo RBF proporciona, en general, buenos resultados en un gran rango de aplicaciones, por este motivo se ha escogido para el desarrollo del proyecto. Es importante decir que estas son las más conocidas y por tanto las más implementadas en las librerías que proporcionan la SVM; pero existen muchos otros tipos de núcleos, tantos como problemas a resolver.

La correcta elección de la función núcleo es muy importante, ya que cada tipo de núcleo crea una estructura diferente para el conjunto de datos de entrada.

I.2.5.4 Fortalezas y debilidades de las SVMs frente a las redes neuronales

Si se realiza una comparación de las SVMs frente a las redes neuronales, de las SVMs se puede afirmar:

- Son más rápidas.
- Su entrenamiento es más sencillo.
- Pueden ser utilizadas con un mayor número de datos en el espacio de entrada.
- Se alcanza la solución óptima (hiperplano óptimo). En las redes neuronales, la solución óptima (mínimo absoluto) no siempre es posible alcanzarlo, siendo un mínimo local la solución alcanzada.
- Generalizan mejor. Al elegir el hiperplano óptimo, el margen es mayor y ante la llegada de muestras nuevas, la probabilidad de que la máquina las clasifique correctamente, aumenta.
- El compromiso entre la complejidad del clasificador y el error puede ser controlado explícitamente.
- Los datos no tradicionales, como cadenas de caracteres y árboles, pueden ser usados como entrada a la SVM, en lugar de vectores de características.

Como debilidad de las SVMs hay que destacar, la elección de una buena función núcleo; es decir, se necesitan metodologías eficientes para sintonizar los parámetros de inicialización de la SVM para que ésta sea eficiente.

PARTE II

CONTROL DE SENSORES EO EN UAVS

UNa vez definida la plataforma sobre la que se va a trabajar, hay que plantear el problema que se quiere resolver. Como bien se ha explicado en la sección I.2.2, una de las misiones principales de los UAVs es el reconocimiento de terrenos, para ello poseen en la parte inferior del fuselaje una cámara con un sensor EO (Electro-Óptico).

Los sensores EO constan de dos componentes, una cámara, la cual se encarga de capturar las imágenes; y un designador láser (LRF, *Laser Range Finder*), cuya función es la medición de distancias láser. Todo esto se monta en una caja robusta y estanca, que dispone de elementos de refrigeración y ha de estar sobrepresurizada con nitrógeno seco para garantizar el máximo rendimiento y fiabilidad de los sensores.

En la imagen 18 pueden verse ejemplos de sensores EO. Una de las compañías suministradoras de EADS de este tipo de sistemas es Cedip, que suministró el sensor EO del UAV DRAC (ver figura I.2.6).

Figura 18: Ejemplos de sensores EO [Sen10].

Los sensores EO, con los cuales se trabajan para los nuevos desarrollos, son capaces de capturar imágenes, tanto en el espectro visible como en el infrarrojo (IR). Es importante resaltar que poseen dos ejes de movimiento, el eje de azimut, el cual permite realizar movimientos en el eje horizontal de la cámara; y el eje de elevación, que permite realizar movimientos en el eje vertical. Son estos dos ejes los que definen la actitud de la cámara y lo que se desea calcular en el algoritmo de geo-apuntamiento.

El problema, al que hay que hacer frente, es el manejo de la cámara que porta en la parte inferior del fuselaje el UAV, para geo-localizar o geo-apuntar a lugares que se está sobrevolando y el operador de la estación de tierra le indique. La cámara posee dos modos de funcionamiento, el operador le puede indicar a la cámara que apunte a unas coordenadas determinadas (geo-apuntamiento) o se le puede preguntar a qué coordenadas está apuntando en ese instante (geo-localización).

Una de las fases más importantes en un proyecto es la definición de requisitos. En este caso, a este proyecto le llegaron como entradas, el problema que había que resolver y los requisitos mínimos que debía cumplir la implementación [DS09].

Los requisitos impuestos para la implementación de los algoritmos son el lenguaje de programación y el entorno de desarrollo en el que se desea implementar la aplicación. Se ha especificado que el entorno de desarrollo sea *Microsoft Visual Studio 2005*, aunque posteriormente se tendrá que modificar y utilizar *Microsoft Visual Studio 6.0* por exigencias de la librería FSUIPC (*Flight Simulator Universal Inter-Process Communication*, ver II.4) para FS (*Microsoft Flight Simulator*). En cuanto al lenguaje de programación, se utilizará C++ [Str00]; esto se decidió así, porque la librería FSUIPC que permite integrar los programas, está implementada para programas en C++.

Como breve introducción a C++, hay que resaltar que nació como una extensión de C, añadiéndole clases, lo cual le dio la característica de orientación a objetos. Además, es un lenguaje compilado, que significa que el fichero en lenguaje máquina se cree antes de la ejecución, en lugar de hacerse en tiempo real como hacen los lenguajes interpretados, esto es una ventaja. Por el contrario, hacen un uso completo de la CPU disponible, que implica la necesidad de una buena gestión de memoria.

Algunas de las características más importantes de C++ son:

- Flexibilidad a la hora de escribir cualquier tipo de programa.
- Eficiencia a la hora de utilización del *hardware*.
- Disponibilidad de existencia de compiladores para la mayoría de las plataformas *hardware*.
- Portabilidad entre máquinas.

Ya que en un futuro se desea que el *software* desarrollado sea posible embarcarlo en los UAVs, todas las restricciones, que sea posible cumplir, se aplicarán a la implementación actual en C++. Para ello, se restringen las funcionalidades del lenguaje. No está permitido utilizar la librería estándar de C++; pero en cambio, si están permitidas algunas funciones de la librería estándar de C. A continuación, se enumeran los ficheros de cabecera de librería que están permitidos usar.

- **float.h**: Define los límites y parámetros del tipo básico *float*.
- **math.h**: Provee las constantes y métodos para realizar operaciones matemáticas básicas, como por ejemplo, funciones trigonométricas, logarítmicas, ...
- **ctype.h**: Contiene las declaraciones de las funciones de clasificación de caracteres.
- **limit.h**: Define los límites de los tipos de datos básicos que ofrece C.
- **stddef.h**: Define las macros NULL, offsetof, ptrdiff_t, wchar_t and size_t.
- **string.h**: Define las clases para manejar cadenas de caracteres.
- **errno.h**: Define las macros para reportar condiciones de error a través de los códigos asociados a errores.
- **signal.h**: Contiene las cabeceras y constantes para el manejo de señales.
- **stdio.h**: Contiene las cabeceras que proveen la interfaz para manejar la entrada y salida de datos (ficheros en disco, en memoria, *streams* y *sockets*). Es la llamada "*standard input/ouput header*"

Como base para la implementación de los algoritmos se ha utilizado un libro de programación en C++ [Str00] y la ayuda que proporciona *Microsoft Visual Studio* [API05].

Los capítulos siguientes presentan el proceso que se ha seguido para la implementación de los algoritmos, la teoría en la que se basan, las pruebas que se les han realizado y la integración con el simulador de vuelo *Microsoft Flight Simulator*.

Capítulo II.1

Sistemas de Referencia

Resumen

El modo de representar un punto sobre una superficie depende del sistema de referencia que se elija. El sistema de referencia está determinado por el origen del mismo y la orientación de los ejes que lo forman. En este capítulo se explican los distintos sistemas de coordenadas que se han utilizado para el desarrollo del presente proyecto.

Existen distintos sistemas para representar unas coordenadas. A continuación, se van a explicar los que se van a utilizar en este proyecto, todos son ortogonales y cartesianos y difieren principalmente en el origen y la orientación de los ejes. La elección de cada uno de ellos depende del movimiento que se desea realizar [[GWP01], [FC01] y [Reu]].

II.1.1 Elipsoide de referencia WGS84

Los sistemas de referencia de ejes geodésicos surgieron por la necesidad de modelar la tierra para poder trabajar con ella, debido a que ésta no es una esfera perfecta. Algunos ejemplos son:

- **WGS84, 72, 64 y 60 del Sistema Geodésico Mundial**
- NAD83, *North American Datum 1983*, es similar al WGS84,
- NAD27, *North American Datum 1927*, es la versión anterior a NAD83
- OSGB36, *Ordnance Survey Great Britain 1936*
- ED50, *European Datum 1950*

El sistema WGS84 (*World Geodetic System 1984*) [WGS00] es el sistema de coordenadas mundiales, que data de 1984, tal y como su nombre indica. Se trata de un sistema de referencia creado por la Agencia de Cartografía del Departamento de Defensa de los Estados Unidos de América (DMA, *Defence Mapping Agency*), para sustentar la cartografía que ellos producían. Es la base para los sistemas de posicionamiento global como GPS (*Global Positioning System*); y además, se utiliza en geodesia, navegación, topografía...

WGS84 es un sistema de coordenadas geodético global, basado originalmente en observaciones *doppler* del sistema de satélites TRANSIT. El elipsoide de referencia de WGS84 es esencialmente el del Sistema Geodésico de referencia 1980 (GRS80, *Geodesic Reference System*), de la Unión Geodésica y Geofísica Internacional, con cambios sólo en su achatamiento.

WGS84 no es sólo un sistema de coordenadas; sino que además, es un sistema de referencia y un modelo gravitacional.

El sistema de coordenadas WGS84 tiene el origen en el centro de masas de la tierra. Los ejes del sistema están establecidos tal y como se muestra en la figura II.1.1. El eje Z_{WGS84} es paralelo a la dirección del polo terrestre convencional (CTP, *Conventional Terrestrial Pole*), el cual fue definido por BIH (*Bureau International de l'Heure*). El meridiano cero es paralelo al BIH. El eje X_1 es la intersección del plano del meridiano de referencia y el CTP ecuatorial. El plano Y_{WGS84} completa el triedro a derechas del sistema de coordenadas cartesiano.

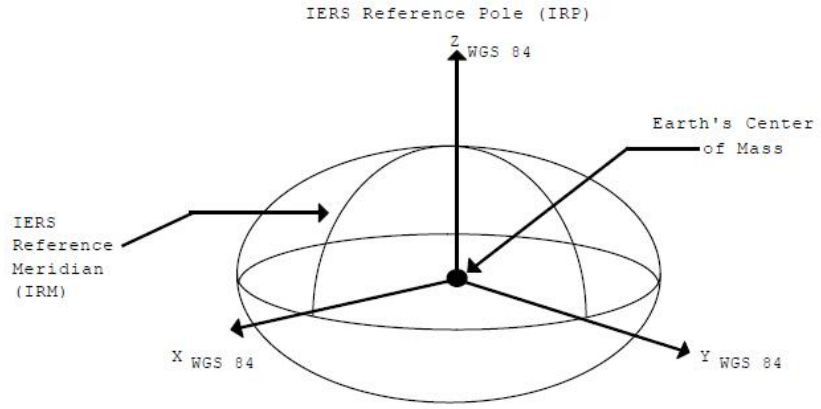


Figura II.1.1: Sistema Geodésico [Los ejes WGS84 son X_{WGS84}, Y_{WGS84} y Z_{WGS84}].

El elipsoide de referencia WGS84 es una elipsoide geocéntrica de revolución. El origen del sistema de coordenadas WGS84 sirve como centro geométrico y los ejes como ejes del elipsoide. Los parámetros del elipsoide WGS84 se muestran a continuación.

- **Semieje mayor**: $a = 6{,}378{,}137m$
- **Semieje menor**: $b = 6{,}356{,}752m$
- **Achatamiento**: $f = \frac{a-b}{a} = \frac{1}{298{,}257223563}$
- **Excentricidad**: $e^2 = \frac{a^2-b^2}{a^2} = 0,00669438$

El sistema coordinado del WGS84 es un sistema de referencia terrestre convencional (CTRS, *Conventional Terrestrial Reference System*), y su definición como sistema coordenado sigue los criterios definidos en la nota técnica 21 del servicio internacional de la rotación de la tierra (IERS, *International Earth Rotation Service*).

II.1.2 Sistema de coordenadas Geodéticas o LLA

El sistema de coordenadas geodéticas o LLA (*Latitude Longitude Altitude*) describe la posición de un punto sobre la superficie terrestre en términos de latitudes, longitudes y alturas geodéticas (ver sección I.2.3).

En la figura II.1.2 puede verse un ejemplo de coordenadas LLA.

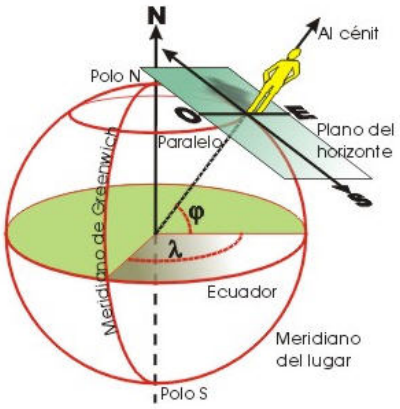


Figura II.1.2: Sistema de coordenadas LLA.

II.1.3 Sistema de coordenadas ECEF

El sistema de coordenadas ECEF (*Earth Centered Earth Fixed*) también es conocido como sistema de coordenadas geocéntrico, *e-frame* o sistema de referencia inercial. Tiene su origen en el centro de masa de la Tierra y sus ejes rotan con ella. Esto hace que el sistema sea no inercial, aunque se considera inercial en muchas aplicaciones de dinámica de vuelo. El eje Z_{EC} se dirige directamente al norte a lo largo del eje polar, paralelo al eje de rotación de la tierra. Los ejes X_{EC} e Y_{EC} están en el plano ecuatorial, con X_{EC} dirigido hacia el meridiano de Greenwich (0° latitud, 0° longitud) e Y_{EC} formando un triedro con el eje X_{EC} y Z_{EC} [ver la figura II.1.3]. Este sistema es útil para referenciar posiciones terrestres.

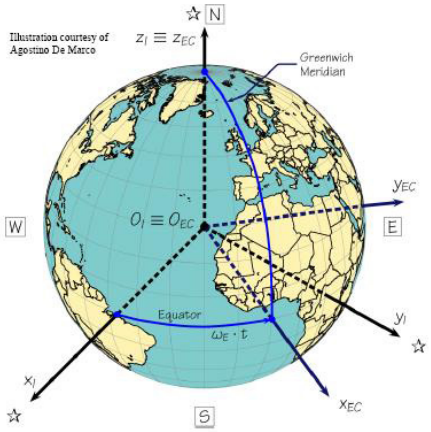


Figura II.1.3: Sistema de coordenadas ECEF.

II.1.4 Sistema de coordenadas NED

El sistema de coordenadas NED (*North East Down*), también conocido como LLS (*Local Level System*), tiene su origen en la localización del sistema inercial. Es un sistema local centrado en un punto, que puede estar en la superficie terrestre o no; por lo tanto, al moverse el punto, el sistema de referencia cambia. Los ejes X_e e Y_e se encuentran en el plano tangente al punto de la Tierra, donde está el origen. El eje X_e apuntará al norte, el eje Y_e al Este y el eje Z_e hacia abajo, tal y como indica su propio nombre.

Otra posible configuración, aunque no es la que se ha utilizado en el proyecto, sería con el eje X_e apuntando al este, el eje Y_e apuntando al norte y el eje Z_e apuntando hacia arriba, también conocido como ENU (*East North Up*).

Se ha elegido la configuración NED, porque los ejes NED coinciden con los ejes BODY, (ver la sección II.1.5) cuando el vehículo está a nivel y orientado hacia el norte.

Es importante resaltar que el sistema NED presenta dificultades cuando se acerca a latitudes cercanas a los polos. En la figura II.1.4 se muestra el sistema de coordenadas.

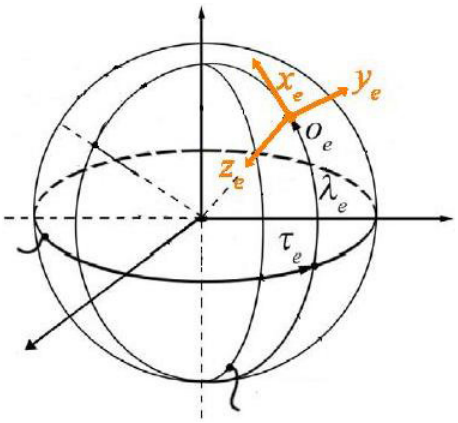


Figura II.1.4: Sistema de coordenadas NED.

II.1.5 Sistema de coordenadas BODY

Este sistema tiene su origen en el centro de masas del vehículo o centro de gravedad. El eje X_B, está en el plano de simetría del vehículo y es positivo hacia adelante; el eje Z_B, es perpendicular al eje X_B y positivo hace abajo en posición normal (por ejemplo en un avión, en la actitud normal de vuelo); y el eje Y_B es el eje que completa el triedro. Normalmente usado en plataformas *Strapdown*; es decir, cuando los sensores tienen también como centro de gravedad el vehículo y sus ejes se mueven con él (es el caso que nos ocupa, ya que las coordenadas de la cámara están dadas con respecto al centro de gravedad del avión).

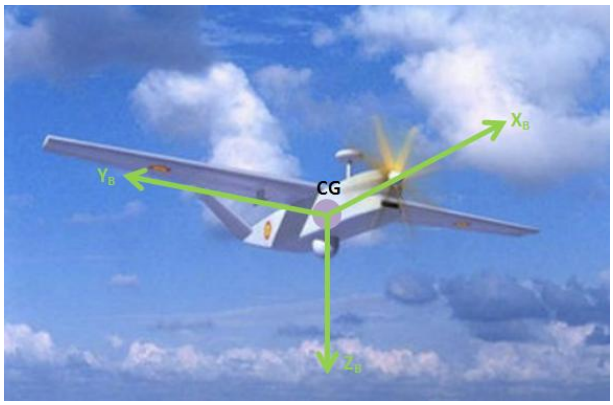


Figura II.1.5: Sistema de coordenadas BODY.

II.1.5.1 Actitud del avión

Se utiliza para definir la actitud y orientación de un avión respecto al sistema de ejes NED. En la figura II.1.6 se ven los tres momentos de un avión, los cuales dan lugar a los ángulos que definen la actitud del avión.

Los ángulos son:

- θ, ángulo de asiento (*pitch*) del vehículo que varía entre [-90° y 90°]. El ángulo de elevación se mide en sentido positivo desde el plano del horizonte local.
- ϕ, ángulo de balance (*roll*) del vehículo que varía entre [-180° y 180°].
- ψ, ángulo de rumbo verdadero (*yaw o heading*) del vehículo que varía entre [-180° y 180°]. El ángulo de Azimut se mide en el sentido de las agujas del reloj desde el norte.

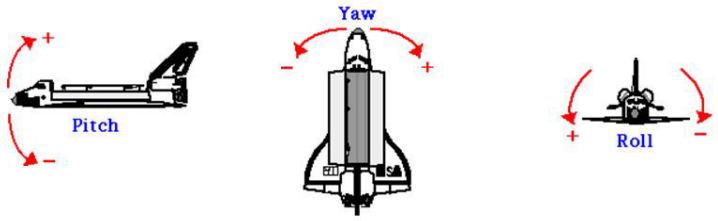


Figura II.1.6: Ángulos del sistema de coordenadas BODY.

II.1.6 Matrices de cambio de coordenadas entre sistemas de referencia

Una vez descritos los sistemas de coordenadas con los que se va a trabajar, es necesario algún tipo de algoritmo para transformar las coordenadas de un sistema a otro. Para ello, en algunos casos se utilizará una formulación específica; y en otros, matrices de cambio de coordenadas.

Las matrices de cambio de coordenadas se basan en los ángulos de rotación que existen entre los dos ejes de coordenadas. El método Matrices de Direcciones Coseno consiste en rotar un vector de las coordenadas origen a las coordenadas destino.

$$C_b^a = \begin{pmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{pmatrix} \tag{II.1.1}$$

Cada componente de esta matriz, es uno de los cosenos de los ángulos entre los ejes de los dos sistemas de coordenadas.

La notación que se va a seguir a lo largo de todo el documento es C_{to}^{from}, que denota la matriz de cambio de coordenadas del sistema de representación "*from* " al sistema de representación "*to*".

Las matrices de transformación de coordenadas cumplen algunas propiedades que son interesantes para este desarrollo.

1. $C_C^A = C_A^B C_B^A$
2. $C_B^A = (C_A^B)^{-1} = (C_A^B)'$

II.1.6.1 Matrices de rotación

Las matrices de rotación son matrices que multiplicadas por el vector de coordenadas del punto, permiten realizar giros en torno a uno de los ejes.

II.1.6.1.1 Giro en $\vec{x}$

Dejando fijo el eje x y sufriendo una variación los ejes y y z, la situación ante la que nos encontramos es la que muestra la figura II.1.7

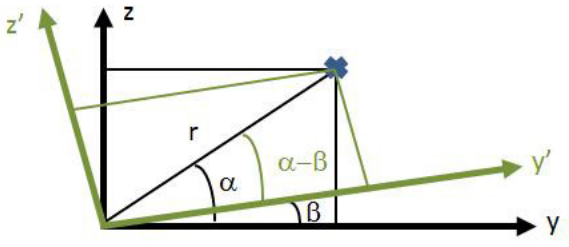


Figura II.1.7: Sistema de coordenadas realizando un giro en el eje x.

Utilizando relaciones trigonométricas para los ángulos α, β y $(\alpha - \beta)$ obtenemos:

$$\cos(\alpha) = \frac{y}{r} \tag{II.1.2}$$

$$\sin(\alpha) = \frac{z}{r} \tag{II.1.3}$$

$\cos(\alpha - \beta) = \cos(\alpha)\cos(\beta) + \sin(\alpha)\sin(\beta) = \frac{y'}{r}$ ➪

$$y' = r\cos(\alpha)\cos(\beta) + r\sin(\alpha)\sin(\beta) = y\cos(\beta) + z\,\mathrm{sen}(\beta) \tag{II.1.4}$$

$\sin(\alpha - \beta) = \sin(\alpha)\cos(\beta) - \cos(\alpha)\sin(\beta) = \frac{z'}{r}$ ➪

$$z' = r\sin(\alpha)\cos(\beta) + r\cos(\alpha)\sin(\beta) = -y\sin(\beta) + z\cos(\beta) \tag{II.1.5}$$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & \sin(\beta) \\ 0 & -\sin(\beta) & \cos(\beta) \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{II.1.6}$$

II.1.6.1.2 Giro en $\vec{y}$

Dejando fijo el eje y y sufriendo una variación los ejes x y z, la situación ante la que nos encontramos es la que muestra la figura II.1.8.

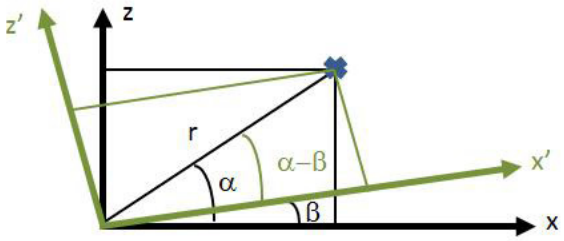


Figura II.1.8: Sistema de coordenadas realizando un giro en el eje y.

Siguiendo la filosofía explicada en el apartado II.1.6.1.1, obtenemos:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos(\beta) & 0 & -\sin(\beta) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{II.1.7}$$

II.1.6.1.3 Giro en $\vec{z}$

Dejando fijo el eje z y sufriendo una variación los ejes x e y, la situación ante la que nos encontramos es la que muestra la figura II.1.9

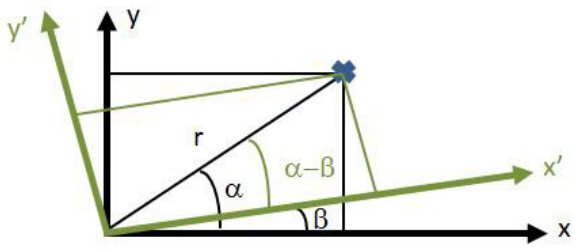


Figura II.1.9: Sistema de coordenadas realizando un giro en el eje z.

Siguiendo la filosofía explicada en el apartado II.1.6.1.1, obtenemos:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos(\beta) & \sin(\beta) & 0 \\ -\sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{II.1.8}$$

II.1.6.2 LLA $\longleftrightarrow$ ECEF

Calculando las coordenadas ECEF a partir de los términos de las ecuaciones I.2.13 y I.2.14 y considerando el eje X en el meridiano cero, se pueden calcular las transformaciones de coordenadas LLA a ECEF y viceversa.

LLA $\longrightarrow$ ECEF

Se calcula el radio de curvatura de la Tierra a una latitud φ dada a partir de los parámetros del elipsoide:

$$N = \frac{a^2}{\sqrt{a^2\cos^2(\varphi) + b^2\sin^2(\varphi)}} \tag{II.1.9}$$

$$X_{ECEF} = \cos(\lambda)\cos(\varphi)(N + h) \tag{II.1.10}$$

$$Y_{ECEF} = \sin(\lambda)\cos(\varphi)(N + h) \tag{II.1.11}$$

$$Z_{ECEF} = \sin(\varphi)\left(\frac{b^2}{a^2} + h\right) \tag{II.1.12}$$

siendo a, el semieje mayor del elipsoide y b, el semieje menor del elipsoide.

LLA $\longleftarrow$ ECEF

Ahora se realizará la transformación contraria, pero primero se definirán algunos parámetros necesarios [1]:

$$\varepsilon = \sqrt{X_{ECEF}^2 + Y_{ECEF}^2} \tag{II.1.13}$$

$$\zeta = atan2(a * Z_{ECEF}, b\varepsilon) \tag{II.1.14}$$

[1] La función $atan2(y, x)$ es la función del cuarto cuadrante en Matlab. En C++ la función devuelve el arcotangente principal de y/x en en el intervalo [-π,π] radianes.

$$r_T = \frac{a}{\sqrt{1 - e^2 \sin^2(\varphi)}} \tag{II.1.15}$$

$$\text{Longitud} = \lambda = \text{atan2}(Y_{ECEF}, X_{ECEF}) \tag{II.1.16}$$

$$\text{Latitud} = \varphi = \text{atan2}\left(Z_{ECEF} + \frac{e^2 a^2 \sin^3(\zeta)}{b}, \varepsilon - e^2 a \cos^3(\zeta)\right) \tag{II.1.17}$$

$$\text{Altura} = h = \frac{\varepsilon}{\cos\varphi} - r_T \tag{II.1.18}$$

siendo a, el semieje mayor del elipsoide; b, el semieje menor del elipsoide; e, la excentricidad del elipsoide y r_T, el radio transversal del elipsoide.

II.1.6.3 ECEF $\longleftrightarrow$ NED

Para calcular la matriz de cambio de coordenadas es necesario tener en cuenta la posición geográfica del punto (longitud (λ) y latitud (φ)) y realizar los siguientes pasos:

- Rotar $\lambda°$ en torno al eje Z_{ECEF}
- Rotar $\varphi°$ en torno al eje Y_{nuevo}
- Rotar -90° para colocar los ejes correctamente ya que el eje X está en la dirección $-Z$ y el eje Z en la dirección X

Lo que obtenemos de estas operaciones son,

$$C_{NED}^{ECEF} = \begin{pmatrix} \cos\lambda & \sin\lambda & 0 \\ -\sin\lambda & \cos\lambda & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{pmatrix} * \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix} \tag{II.1.19}$$

$$C_{NED}^{ECEF} = \begin{pmatrix} -\cos\lambda\sin\varphi & -\sin\lambda\sin\varphi & \cos\varphi \\ -\sin\lambda & \cos\lambda & 0 \\ -\cos\lambda\cos\varphi & -\sin\lambda\cos\varphi & -\sin\varphi \end{pmatrix} \tag{II.1.20}$$

Una vez calculada la matriz de cambio de ECEF a NED, la matriz de transformación de NED a ECEF se consigue de manera inmediata transponiendo la matriz.

$$C_{ECEF}^{NED} = C_{NEC}^{ECEF^T} = \begin{pmatrix} -\cos\lambda\sin\varphi & -\sin\lambda & -\cos\lambda\cos\varphi \\ -\sin\lambda\sin\varphi & \cos\lambda & -\sin\lambda\cos\varphi \\ \cos\varphi & 0 & -\sin\varphi \end{pmatrix} \tag{II.1.21}$$

II.1.6.4 NED $\longleftrightarrow$ BODY

Para calcular la matriz de cambio de coordenadas, es necesario tener en cuenta la actitud del avión: ángulo de *pitch* (P), ángulo de *roll* (R) y ángulo de *heading* (H). Después hay que realizar los siguientes pasos,

- Rotar H° en torno al eje Z_{ECEF}.
- Rotar P° en torno al nuevo eje Y_{ECEF}.
- Rotar R° en torno al nuevo eje X_{ECEF}.

Lo que obtenemos de estas operaciones son,

$$C_{BODY}^{NED} =$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos R & \sin R \\ 0 & -\sin R & \cos R \end{pmatrix} * \begin{pmatrix} \cos P & 0 & \sin P \\ 0 & 1 & 0 \\ -\sin P & 0 & \cos P \end{pmatrix} * \begin{pmatrix} \cos H & \sin H & 0 \\ -\sin H & \cos H & 0 \\ 0 & 0 & 1 \end{pmatrix} = \quad \text{(II.1.22)}$$

$$= \begin{pmatrix} \cos P \cos H & \cos P \sin H & -\sin P \\ \sin R \sin P \cos H - \cos R \sin H & \sin H \sin P \sin R + \cos R \cos H & \cos P \sin R \\ \cos R \sin P \cos H + \sin R \sin H & \sin H \sin P \cos R - \sin R \cos H & \cos P \cos R \end{pmatrix} \quad \text{(II.1.23)}$$

Una vez calculada la matriz de cambio de NED a BODY, la matriz de transformación de BODY a NED se consigue de manera inmediata transponiendo la matriz.

$$C_{NED}^{BODY} = C_{BODY}^{NED\ T} =$$

$$= \begin{pmatrix} \cos P \cos H & \sin R \sin P \cos H - \cos R \sin H & \cos R \sin P \cos H + \sin R \sin H \\ \cos P \sin H & \sin H \sin P \sin R + \cos R \cos H & \sin H \sin P \cos R - \sin R \cos H \\ -\sin P & \cos P \sin R & \cos P \cos R \end{pmatrix} \quad \text{(II.1.24)}$$

II.1.7 Coordenadas y matrices homogéneas

Existen cuatro operaciones básicas que pueden realizarse en sistema de tres ejes, como las que se van a utilizar en este proyecto: la rotación, la traslación, la reflexión y el escalado. En este proyecto únicamente se va a "jugar"con los movimientos de traslación y rotación de los ejes.

La rotación consiste en girar los ejes con respecto a una referencia. En este caso, los giros se realizan con respecto a alguno de los ejes del sistema, explicados en la sección II.1.6.1.

La traslación consiste en mover el origen de los ejes de coordenadas a lo largo de sus ejes. Para ello, únicamente es necesario sumar la cantidad que se quiere trasladar en cada uno de los ejes.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x + d_x \\ y + d_y \\ z + d_z \end{pmatrix} \tag{II.1.25}$$

Cuando se trabaja con matrices de rotación, éstas no permiten realizar traslaciones. Por este motivo se va a recurrir al concepto de coordenadas y matrices homogéneas [[Dep] [Vel] y [vDH97]]. Si se utilizan las matrices homogéneas para representar los ejes de coordenadas, las tres operaciones pueden ser tratadas con una multiplicación de matrices.

Las coordenadas homogéneas fueron desarrolladas por August Ferdinand Möbius en 1837. La aplicación más extendida es la geometría proyectiva. Las coordenadas nos permiten representar un punto en un espacio proyectivo. También pueden usarse como un sistema alternativo de coordenadas para trabajar en el espacio euclídeo, pues éste puede verse como un subconjunto del espacio proyectivo.

La idea de las coordenadas homogéneas, consiste en introducir en un sistema de tres coordenadas, una cuarta coordenada (φ). Esta nueva coordenada representa un factor de escala, como norma general, este valor suele ser 1 y en este caso las tres primeras componentes de las coordenadas homogéneas y normales coinciden.

$$\begin{pmatrix} x \\ y \\ z \\ \varphi \end{pmatrix} = \begin{pmatrix} x\varphi \\ y\varphi \\ z\varphi \\ \varphi \end{pmatrix} \tag{II.1.26}$$

La matriz 4x4 que representa la transformación de un vector de coordenadas homogéneas de un sistema de coordenadas a otro, realizando las operaciones de rotación y traslación es:

$$T = \begin{pmatrix} \text{Rotación}_{3x3} & \text{Traslación}_{3x1} \\ \text{Reflexión}_{1x3} & \text{Escalado}_{1x1} \end{pmatrix} = \begin{pmatrix} \text{Rotación}_{3x3} & \text{Traslación}_{3x1} \\ 0 & 1 \end{pmatrix} \tag{II.1.27}$$

A continuación se muestran las matrices homogéneas que permiten realizar traslaciones y rotaciones. Para ello, basta con multiplicar las matrices por el vector, en coordenadas homogéneas.

Matriz de traslación de una distancia $d = (dx, dy, dz)$,

$$T_d = \begin{pmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{II.1.28}$$

Giro de αř en cada uno de los ejes X, Y y Z,

$$GX_\alpha = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{II.1.29}$$

$$GY_\alpha = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{II.1.30}$$

$$GZ_\alpha = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{II.1.31}$$

Capítulo II.2

Algoritmos para el control de los sensores EO

Resumen

En este capítulo, se explican los algoritmos de geo-localización y geo-apuntamiento que se van a implementar en esta parte del proyecto.

UNa vez revisados los conceptos más importantes de navegación, definición de tipos de coordenadas y cambios entre ellas, se pueden comenzar a estudiar los algoritmos que se quieren implementar [DS09]. Antes de realizar esto, es necesario caracterizar la cámara que se va a utilizar para el desarrollo de los algoritmos.

II.2.1 Caracterización del sensor EO del UAV

El sensor EO que porta el UAV está definido por el vector LOS (Line Of Sigth), formado por el ángulo de azimut, el ángulo de elevación y la distancia al objetivo; y por el vector de traslación del sensor EO con respecto al centro de gravedad del avión.

- $\beta \Rightarrow$ Ángulo de azimut (*panoramic*) de la línea de vista del sensor EO, respecto a los ejes del cuerpo.
- $\alpha \Rightarrow$ Ángulo de elevación (*tilt*) de la línea de vista del sensor EO, respecto a los ejes del cuerpo.

Para el desarrollo de este proyecto, se ha tomado como decisión de diseño, que todos los parámetros del sensor estén referenciados a los ejes del avión. En la figura II.2.1 pueden verse estos conceptos.

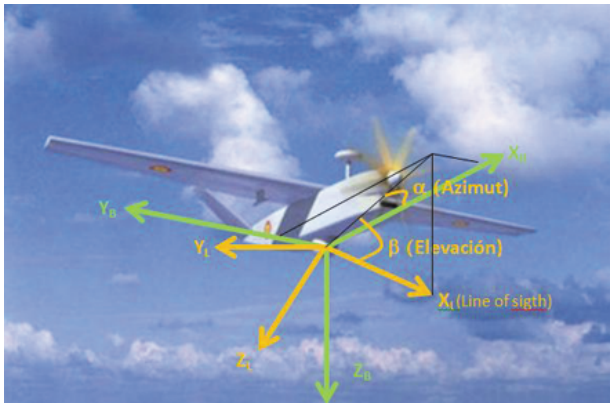


Figura II.2.1: Sistema de coordenadas LOS.

II.2.1.1 Cálculo de los ángulos Azimut y Elevación a partir de los ejes BODY del avión

Se van a desarrollar las ecuaciones que permiten calcular los ángulos de azimut y elevación de la cámara, a partir de las coordenadas en ejes BODY. Para una mejor comprensión ver la figura II.2.1.

$$\boxed{Azimut = \alpha = atan2(Y_{BODY}, X_{BODY})} \qquad \text{(II.2.1)}$$

$$H = hipotenusa = X^2_{BODY} + Y^2_{BODY} = \cos(\beta)|X_{LOS}| \tag{II.2.2}$$

$$X_{BODY} = \cos(\alpha) * H \tag{II.2.3}$$

$$Z_{BODY} = \sin(\beta) * |X_{LOS}| \tag{II.2.4}$$

Sustituyendo

$$\boxed{Elevación = \beta = atan2(Z_{BODY}, \sqrt{X^2_{BODY} + Y^2_{BODY}})} \tag{II.2.5}$$

II.2.2 Algoritmo para el cálculo de la posición de un objetivo (Geo-localización)

La imagen II.2.2 ilustra gráficamente el algoritmo de geo-localización que se va a implementar.

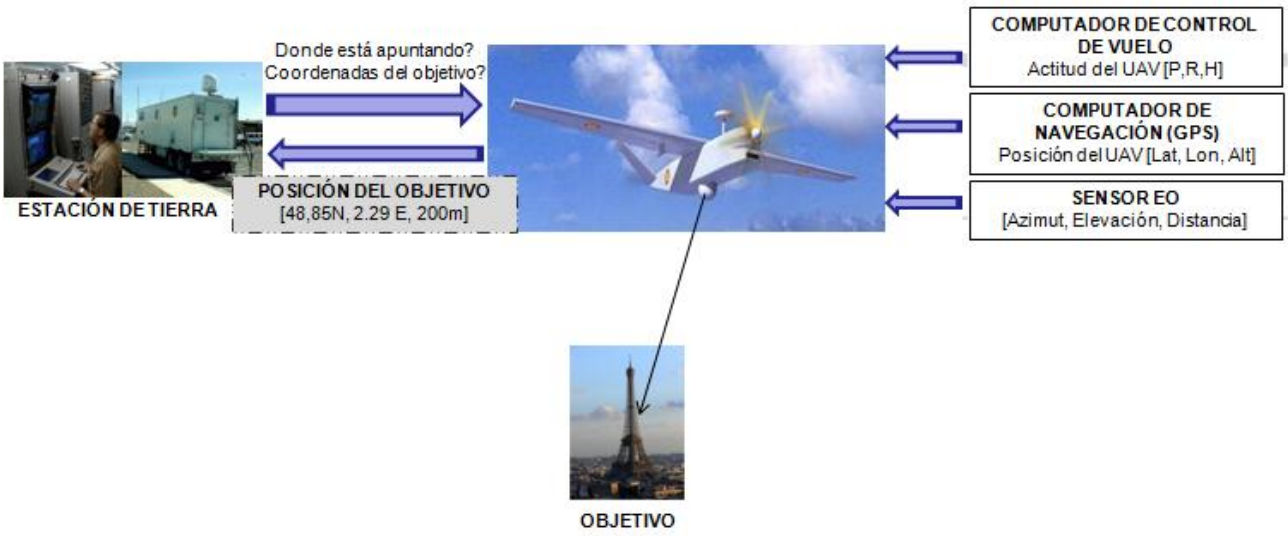


Figura II.2.2: Algoritmo de geo-localización.

Como se puede ver en la imagen II.2.2, los datos de entrada al algoritmo son:

- **Posición del avión**: longitud (λ_{AC}), latitud (φ_{AC}) y altura (h_{AC}).
- **Actitud del avión**: ángulos de *pitch*, *roll* y *yaw*.
- **Actitud de la cámara**: ángulos de azimut (α) y elevación (β).
- **Distancia al objetivo**: dada por el designador láser del sensor EO ($|LOS|$).
- **Vector de traslación del sensor EO con respecto a los ejes del avión** ($\overrightarrow{T_{EO\_CG}}$).

El cálculo de la posición geográfica del objetivo designado en el sensor EO del UAV, está basado en el cálculo del vector $\overrightarrow{TGT}$ en el sistema de coordenadas ECEF, tal y como muestra la figura II.2.3.

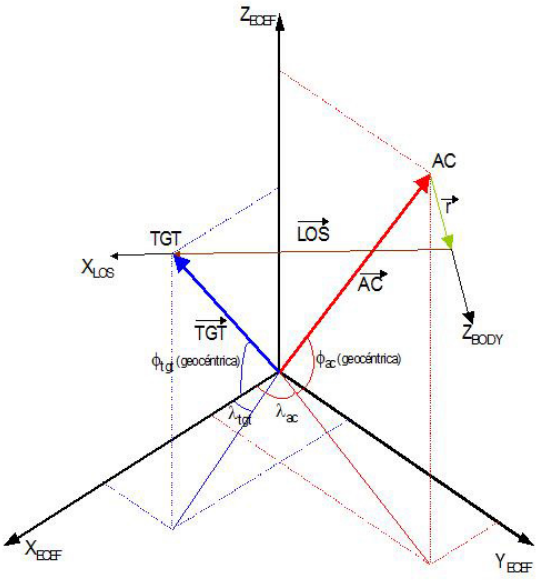


Figura II.2.3: Cálculo de la posición de un TGT (Geo-localización).

Los pasos a seguir se detallan a continuación:

Paso 1: Definición del vector $\overrightarrow{AC_{LLA}}$. Este vector caracteriza la posición del avión en coordenadas LLA (ver II.1.2).

$$\overrightarrow{AC_{LLA}} = \begin{pmatrix} \lambda_{AC} \\ \varphi_{AC} \\ h_{AC} \end{pmatrix} \tag{II.2.6}$$

Paso 2: Cambio de coordenadas del vector $\overrightarrow{AC_{LLA}}$ a coordenadas ECEF (ver II.1.6.2 y II.1.3). Con este cambio se obtiene el vector:

$$\overrightarrow{AC_{ECEF}} = \begin{pmatrix} X_{AC\_ECEF} \\ Y_{AC\_ECEF} \\ Z_{AC\_ECEF} \end{pmatrix} \tag{II.2.7}$$

Paso 3: Definición del vector $\overrightarrow{LOS_{BODY}}$ en coordenadas BODY. Es el vector que une el centro del sistema BODY con el objetivo que está siendo visualizado y sobre el que se medirá la distancia usando el telémetro láser. Observando la figura II.2.1 y aplicando trigonometría se puede deducir:

$$X_{LOS\_BODY} = \sin(\alpha)|LOS|\cos(\beta) \tag{II.2.8}$$
$$Y_{LOS\_BODY} = \cos(\alpha)|LOS|\cos(\beta) \tag{II.2.9}$$
$$Z_{LOS\_BODY} = \sin(\beta)|LOS| \tag{II.2.10}$$
$$\tag{II.2.11}$$

Paso 4: Trasladar el origen del vector $\overrightarrow{LOS_{BODY}}$ al centro de gravedad del avión (centro de coordenadas de los ejes BODY). Para ello, basta con restar al vector $\overrightarrow{LOS_{BODY}}$, el vector de traslación del sensor EO ($\overrightarrow{T_{EO\_CG}}$).

$$\overrightarrow{LOS_{LOCAL\_BODY}} = \begin{pmatrix} X_{LOS\_BODY} \\ Y_{LOS\_BODY} \\ Z_{LOS\_BODY} \end{pmatrix} - \overrightarrow{T_{EO\_CG}} = \begin{pmatrix} X_{LOS\_BODY} \\ Y_{LOS\_BODY} \\ Z_{LOS\_BODY} \end{pmatrix} - \begin{pmatrix} X_{EO\_CG} \\ Y_{EO\_CG} \\ Z_{EO\_CG} \end{pmatrix} \tag{II.2.12}$$

Paso 5: Transformar el vector $\overrightarrow{LOS_{LOCAL\_BODY}}$ de coordenadas BODY, a coordenadas LOCALES (coordenadas ECEF referidas al punto donde se encuentra el avión). Para realizar este cambio, es necesario el uso de las matrices de cambio C_{NED}^{BODY} y C_{LOCAL}^{NED} definidas en las secciones II.1.6.4 y II.1.6.3. Aplicando las matrices se obtiene el vector $\overrightarrow{LOS_{LOCAL}}$.

$$\overrightarrow{LOS_{LOCAL}} = C_{LOCAL}^{NED} C_{NED}^{BODY} \overrightarrow{LOS_{LOCAL\_BODY}} = \begin{pmatrix} X\_LOS_{LOCAL} \\ Y\_LOS_{LOCAL} \\ Z\_LOS_{LOCAL} \end{pmatrix} \tag{II.2.13}$$

Paso 6: Calcular el vector TGT en coordenadas ECEF. El vector $\overrightarrow{TGT_{ECEF}}$ es el que contiene la posición del objetivo en coordenadas ECEF (ver II.1.2). Para calcularlo, basta con sumar los vectores $\overrightarrow{AC_{ECEF}}$ y $\overrightarrow{LOS_{LOCAL}}$

$$\overrightarrow{TGT_{ECEF}} = \overrightarrow{AC_{ECEF}} + \overrightarrow{LOS_{LOCAL}} = \begin{pmatrix} X\_TGT_{ECEF} \\ Y\_TGT_{ECEF} \\ Z\_TGT_{ECEF} \end{pmatrix} \tag{II.2.14}$$

Paso 7: Calcular las coordenadas LLA a partir del vector $\overrightarrow{TGT_{ECEF}}$. Para ello, es necesario aplicar las fórmulas desarrolladas en la sección II.1.6.2.

$$\varepsilon = \sqrt{X_{ECEF}^2 + Y_{ECEF}^2}; \quad \zeta = atan2(a * Z_{ECEF}, b\varepsilon); \quad r_T = \frac{a}{\sqrt{1 - e^2 \sin^2(\varphi)}}$$

$$\boxed{Longitud_{TGT} = \lambda_{TGT} = atan2(Y\_TGT_{ECEF}, X\_TGT_{ECEF})} \tag{II.2.15}$$

$$\boxed{Latitud_{TGT} = \varphi_{TGT} = atan2(Z_{ECEF} + \frac{e^2 a^2 \sin^3(\zeta)}{b}, \varepsilon - e^2 a \cos^3(\zeta))} \tag{II.2.16}$$

$$\boxed{Altitud_{TGT} = h_{TGT} = \frac{\varepsilon}{\cos\phi} - r_T} \tag{II.2.17}$$

II.2.3 Algoritmo para el apuntamiento del sensor EO a un objetivo (geo-apuntamiento)

La imagen II.2.4 ilustra gráficamente el algoritmo de geo-apuntamiento que se va a implementar.

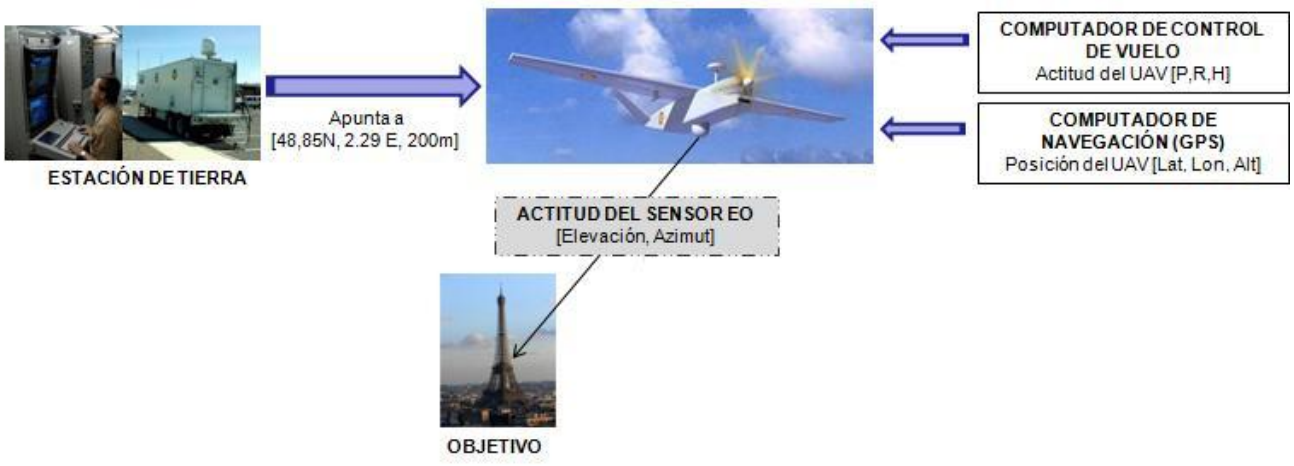


Figura II.2.4: Algoritmo de geo-apuntamiento.

El apuntamiento del sensor EO, está basado en el cálculo del vector $\overrightarrow{LOS}$ en el sistema de coordenadas BODY. A partir de él se calculan los ángulos de elevación (α, *Tilt*) y azimut (β, *Panoramic*) del sensor (ver II.2.3), que establecen la actitud de la cámara.

Los datos de entrada al algoritmo son:

- **Posición del avión**: longitud (λ_{AC}), latitud (φ_{AC}) y altura (h_{AC}).
- **Actitud del avión**: ángulos de *pitch*, *roll* y *yaw*.
- **Posición del objetivo**: longitud (λ_{TGT}), latitud (φ_{TGT}) y altura (h_{TGT}).
- **Vector de traslación del sensor EO con respecto a los ejes del avión** ($\overrightarrow{T_{EO\_CG}}$).

Estos son los pasos a seguir para realizar los cálculos:

Paso 1: Definición del vector $\overrightarrow{AC_{LLA}}$. Este vector caracteriza la posición del avión en coordenadas LLA, definidas en la sección II.1.2.

$$\overrightarrow{AC_{LLA}} = \begin{pmatrix} \lambda_{AC} \\ \varphi_{AC} \\ h_{AC} \end{pmatrix} \tag{II.2.18}$$

Paso 2: Definición del vector $\overrightarrow{TGT_{LLA}}$. Este vector caracteriza la posición del objetivo en coordenadas LLA, definidas en la sección II.1.2.

$$\overrightarrow{TGT_{LLA}} = \begin{pmatrix} \lambda_{TGT} \\ \varphi_{TGT} \\ h_{TGT} \end{pmatrix} \tag{II.2.19}$$

Paso 3: Cambio de coordenadas de los vectores $\overrightarrow{AC_{LLA}}$ y $\overrightarrow{TGT_{LLA}}$ a coordenadas ECEF (ver II.1.6.2 y II.1.3). Con este cambio se consiguen los vectores:

$$\overrightarrow{AC_{ECEF}} = \begin{pmatrix} X_{AC\_ECEF} \\ Y_{AC\_ECEF} \\ Z_{AC\_ECEF} \end{pmatrix} \quad \overrightarrow{TGT_{ECEF}} = \begin{pmatrix} X_{TGT\_ECEF} \\ Y_{TGT\_ECEF} \\ Z_{TGT\_ECEF} \end{pmatrix} \tag{II.2.20}$$

Paso 4: Cálculo del vector $\overrightarrow{LOS}$ en coordenadas ECEF locales, a partir de ahora, coordenadas LOCALES. El vector $\overrightarrow{LOS_{LOCAL}}$ une el centro del sistema LOS con el objetivo que está siendo visualizado y sobre el que se medirá la distancia usando el telémetro láser. Este vector sólo tiene componente según el eje X del sistema de coordenadas LOS y su módulo es igual a la distancia AC-TGT (|LOS|) medida por el telémetro láser.

$$\overrightarrow{LOS_{LOCAL}} = \overrightarrow{TGT_{ECEF}} - \overrightarrow{AC_{ECEF}} \tag{II.2.21}$$

Paso 5: Transformación a coordenadas BODY del vector $\overrightarrow{LOS_{LOCAL}}$. Para ello, se utilizarán las matrices de cambio C^{ECEF}_{NED} y C^{NED}_{BODY} definidas en las secciones II.1.6.3 y II.1.6.4. Aplicando las matrices, obtenemos el vector $\overrightarrow{LOS_{BODY}}$ (ver II.1.5).

$$\overrightarrow{LOS_{BODY}} = C^{NED}_{BODY} C^{ECEF}_{NED} \overrightarrow{LOS_{LOCAL}} = \begin{pmatrix} X_{LOS\_BODY} \\ Y_{LOS\_BODY} \\ Z_{LOS\_BODY} \end{pmatrix} \tag{II.2.22}$$

Paso 6: Es necesario desplazar el centro de los ejes BODY, del centro de gravedad a la posición de la cámara. Para ello, basta con sumar al vector $\overrightarrow{LOS_{BODY}}$ al vector de traslación del sensor EO ($\overrightarrow{T_{EO\_CG}}$).

$$\overrightarrow{LOS} = \begin{pmatrix} X_{LOS\_BODY} \\ Y_{LOS\_BODY} \\ Z_{LOS\_BODY} \end{pmatrix} + \overrightarrow{T_{EO\_CG}} = \begin{pmatrix} X_{LOS\_BODY} \\ Y_{LOS\_BODY} \\ Z_{LOS\_BODY} \end{pmatrix} \begin{pmatrix} X_{EO\_CG} \\ Y_{EO\_CG} \\ Z_{EO\_CG} \end{pmatrix} \tag{II.2.23}$$

Paso 7: Cálculo de los ángulos de azimut y elevación a partir del vector $\overrightarrow{LOS}$.

$$\boxed{Azimut = \alpha = atan2(Y_{LOS\_BODY}, X_{LOS\_BODY})} \tag{II.2.24}$$

$$\boxed{Elevación = \beta = atan2(Z_{LOS\_BODY}, \sqrt{X^2_{LOS\_BODY} + Y^2_{LOS\_BODY}})} \tag{II.2.25}$$

Capítulo II.3

Proyección cónica conforme de Lambert

Resumen

En este capítulo, se explica la proyección cónica conforme de Lambert, que es una forma de representación de la esfera terrestre. Esta proyección permite convertir un punto representado en coordenadas cartesianas, a su representación en coordenadas LLA, y viceversa.

Desde los tiempos antiguos, los humanos conocen la forma esférica y no plana del globo terrestre. Si la tierra fuera plana, la cartografía sería mucho más sencilla porque no sería necesaria la utilización de las proyecciones.

La representación cartográfica del globo terrestre, ya sea considerado como una esfera o como un elipsoide, es un problema; ya que no permite representar toda la superficie del globo sin ser deformada o sin que ésta sea una representación fiel. Esta representación se llama proyección [[FC01] y [P.S87]].

Una proyección es una función que relaciona las coordenadas de un punto en una superficie curva (normalmente una esfera o un elipsoide), con las coordenadas del punto en un plano. La proyección puede ser establecida por análisis computacional o por construcción geométrica, siendo esta última forma menos común.

Todas la proyecciones que se han desarrollado conllevan una distorsión de una o varias propiedades espaciales del globo. Dependiendo de las propiedades que no se deseen distorsionar, fijarán la proyección que se deba utilizar. En la figura (II.3.1) se puede ver el ejemplo de una proyección.

Figura II.3.1: Ejemplo de una proyección.

Las cinco características susceptibles de ser distorsionadas en la proyección del globo son: forma, distancia, dirección, escala y área. Ninguna proyección conserva más de una propiedad de las anteriores, en porciones grandes de la tierra. Esto no se debe a la poca eficiencia de la proyección, si no a la imposibilidad de ser realizado. Los tipos de proyecciones de acuerdo a estas cinco características son:

Conforme o Ortomórfica: esta proyección mantiene la forma local (áreas pequeñas), cuando la escala del mapa en cualquier punto es la misma en cualquier dirección. En estas proyecciones, los meridianos y paralelos se interseccionan con los ángulos correctos.

Equidistante: esta proyección preserva las distancias, únicamente, desde el centro de la proyección a todos los lugares de ésta. Los mapas, únicamente, se describen como equidistantes cuando la separación entre paralelos es uniforme. No hay ninguna proyección que sea capaz de mantener las distancias desde cualquier punto del mapa.

Azimutal: esta proyección preserva la dirección cuando los azimuts son representados correctamente en todas las direcciones.

Equivalente: esta proyección mantiene la escala entre la distancia representada en el mapa y la misma medida en la Tierra. Ninguna proyección mantiene constante la escala en áreas grandes, pero algunas son capaces de limitarla. Un mapa no puede ser equivalente y conforme a la vez.

Ahora es necesario escoger el tipo de proyección más adecuado a la zona que se quiere tratar. Estas proyecciones pueden dividirse en:

- **Proyecciones planas**: son proyecciones representadas en un plano.
- **Proyecciones geodésicas**: son proyecciones en las que la esfericidad terrestre repercute en: la representación de las coordenadas geográficas de un punto, las superficies, los ángulos y las distancias. Dentro de este grupo, las proyecciones pueden clasificarse en función de muchos factores. En este documento, se ha decidido utilizar la clasificación en función de la forma geométrica utilizada para la transformación del globo terráqueo a un plano:
  - **Proyecciones cilíndricas**
  - **Proyecciones cónicas**
  - **Proyecciones azimutal**

En la figura II.3.2 se pueden ver tres ejemplos, uno de cada tipo de proyección.

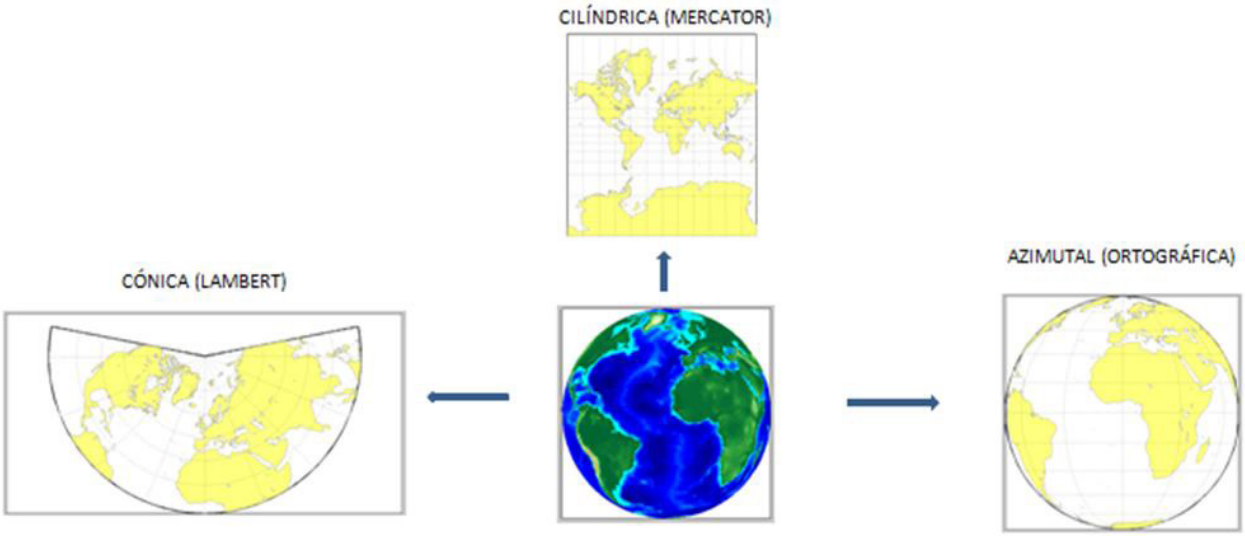


Figura II.3.2: Ejemplos de proyecciones cilíndricas, cónica y azimutal.

Este proyecto se centra en la proyección cónica conforme de Lambert. Se puede ver en la figura II.3.2 como ejemplo de proyección cónica, tal y como su nombre indica.

La proyección cónica conforme de Lambert fue creada por el matemático alsaciano Johann Heinrich Lambert en el año 1772. Ha sido utilizada para el mapeo de las regiones con predominio de orientación este-oeste y en navegación aérea.

El objetivo de la proyección es la representación en un plano de la superficie terrestre de referencia, WGS84 (ver II.1). Para ello, todos los puntos de la superficie de referencia, en este caso elipsoidal, se proyectan sobre un cono tangente a la superficie terrestre. Este cono se desarrolla posteriormente, produciendo la representación plana de la superficie de referencia proyectada (ver figura II.3).

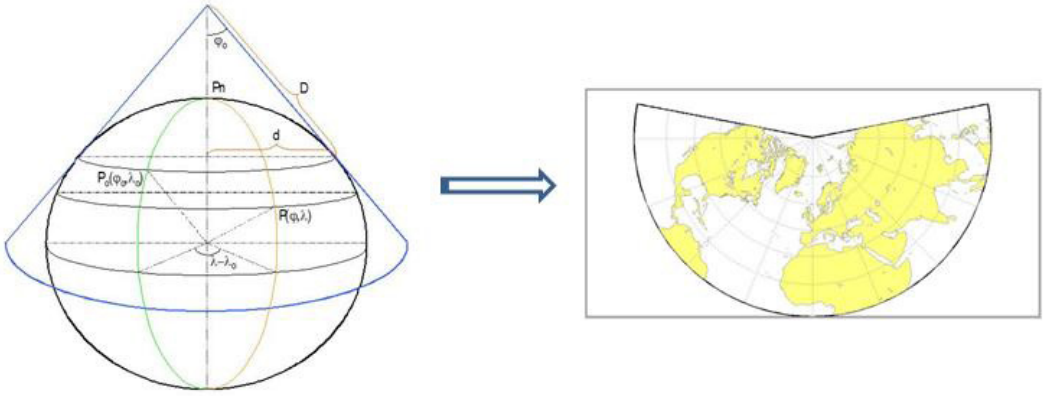


Figura II.3.3: Proyección cónica conforme de Lambert.

En esencia, la proyección superpone un cono sobre la esfera de la Tierra, con dos paralelos de referencia secantes al globo e intersecándolo. Esto minimiza la distorsión producida al proyectar una superficie tridimensional en una bidimensional. La distorsión es mínima a lo largo de los paralelos de referencia, y se incrementa fuera de los paralelos elegidos. Como el nombre lo indica, esta proyección es conforme.

Sobre la base de la proyección cónica simple, con dos meridianos de referencia, Lambert ajustó matemáticamente la distancia ente paralelos para crear un mapa conforme. Como los meridianos son líneas rectas y los paralelos arcos de círculo concéntricos, las diferentes hojas encajan perfectamente.

Los pilotos utilizan estas cartas, debido a que una línea recta dibujada sobre una carta cuya proyección es cónica conforme de Lambert, muestra la distancia verdadera entre puntos. Sin embargo, los aviones deben volar rutas que son arcos de círculos máximos, para recorrer la distancia más corta entre dos puntos de la superficie. En una carta de Lambert, estos puntos aparecen como una línea curva, que debe ser calculada de forma separada, para asegurar la identificación de los puntos intermedios correctos en la navegación.

Las características más importantes son:

- Los meridianos se trasforman en rectas concurrentes en un punto. Este punto de concurrencia es el trasformado del polo de la proyección.
- Los paralelos se trasforman en arcos de circunferencia concéntricos, cuyo centro es el punto de concurrencia de los meridianos.
- Al ser una proyección conforme, los ángulos se conservan, es decir, los ángulos medidos en la proyección, se corresponden con los ángulos en la superficie de referencia proyectada.
- El paralelo en el cual el cono es tangente, es llamado paralelo automecóico. Es un paralelo en el que no existen deformaciones lineales. La relación lineal entre la referencia terrestre y la proyección, esta afectada únicamente por la escala.
- La curva ortodrómica se transforma en una recta en el plano.

II.3.1 Transformación directa para elipsoides

Dados los siguientes parámetros de entrada,

- a: Semieje mayor del elipsoide WGS84.
- e: Excentricidad del elipsoide WGS84.
- φ_1: Latitud del paralelo estándar (se considera φ_0+1°).
- φ_2: Latitud del paralelo estándar (se considera φ_0-1°).
- φ_0: Latitud del origen.
- λ_0: Longitud del origen.
- φ: Latitud del punto.
- λ: Longitud del punto.

las fórmulas para calcular las coordenadas cartesianas a partir de las coordenadas LLA del punto se muestran en la tabla II.3.1.

$\theta = n(\lambda - \lambda_0)$	$F = \frac{m_1}{(nt_1^n)}$	$\rho = aFt^n$	$\rho_0 = aFt_0^n$
$k = \frac{\rho * n}{a * m} = \frac{m_1 t^n}{m * t_1^n}$		$n = \frac{\ln m_1 - \ln m_2}{\ln t_1 - \ln t_2}$	
$m = \frac{\cos\varphi}{(1-e^2\sin^2\varphi)^{\frac{1}{2}}}$	$m_0 = \frac{\cos\varphi_0}{(1-e^2\sin^2\varphi_0)^{\frac{1}{2}}}$	$m_1 = \frac{\cos\varphi_1}{(1-e^2\sin^2\varphi_1)^{\frac{1}{2}}}$	$m_2 = \frac{\cos\varphi_2}{(1-e^2\sin^2\varphi_2)^{\frac{1}{2}}}$
$t = \frac{\tan\frac{\pi}{4}-\frac{\varphi}{2}}{\left[\frac{(1-e\sin\varphi)}{(1+e\sin\varphi)}\right]^{\frac{1}{2}}}$	$t_0 = \frac{\tan\frac{\pi}{4}-\frac{\varphi_0}{2}}{\left[\frac{(1-e\sin\varphi_0)}{(1+e\sin\varphi_0)}\right]^{\frac{1}{2}}}$	$t_1 = \frac{\tan\frac{\pi}{4}-\frac{\varphi_1}{2}}{\left[\frac{(1-e\sin\varphi_1)}{(1+e\sin\varphi_1)}\right]^{\frac{1}{2}}}$	$t_2 = \frac{\tan\frac{\pi}{4}-\frac{\varphi_2}{2}}{\left[\frac{(1-e\sin\varphi_2)}{(1+e\sin\varphi_2)}\right]^{\frac{1}{2}}}$

Tabla II.3.1: Formulación para el cálculo de las coordenadas cartesianas a partir de las coordenadas LLA.

$$\boxed{x = \rho \sin\theta} \tag{II.3.1}$$

$$\boxed{y = \rho_0 - \rho\sin\theta} \tag{II.3.2}$$

II.3.2 Transformación indirecta para elipsoides

Dados los siguientes parámetros

- a: Semieje mayor del elipsoide WGS84.
- e: Excentricidad del elipsoide WGS84.
- φ_1: Latitud del paralelo estándar (se considera φ_0+1°).
- φ_2: Latitud del paralelo estándar (se considera φ_0-1°).
- φ_0: Latitud del origen.
- λ_0: Longitud del origen.
- x: Coordenada x del punto.
- y: Coordenada y del punto.

las fórmulas para calcular las coordenadas LLA a partir de las coordenadas cartesianas del punto se muestran en la tabla II.3.2

$\theta = \arctan\frac{x}{\rho_0 - y}$	$F = \frac{m_1}{(nt_1^n)}$	$\rho_0 = aFt_0^n$	$\rho = \sqrt{x^2 + (\rho_0 - y)^2}$
$k = \frac{\rho * n}{a * m} = \frac{m_1 t^n}{m * t_1^n}$		$n = \frac{\ln m_1 - \ln m_2}{\ln t_1 - \ln t_2}$	
Si n es negativo, los signos de ρ, ρ_0,x e y deben ser invertidos			
	$m_0 = \frac{\cos\varphi_0}{(1-e^2\sin^2\varphi_0)^{\frac{1}{2}}}$	$m_1 = \frac{\cos\varphi_1}{(1-e^2\sin^2\varphi_1)^{\frac{1}{2}}}$	$m_2 = \frac{\cos\varphi_2}{(1-e^2\sin^2\varphi_2)^{\frac{1}{2}}}$
$t = \sqrt[n]{\frac{\rho}{aF}}$	$t_0 = \frac{\tan\frac{\pi}{4} - \frac{\varphi_0}{2}}{\left[\frac{(1-e\sin\varphi_0)}{(1+e\sin\varphi_0)}\right]^{\frac{1}{2}}}$	$t_1 = \frac{\tan\frac{\pi}{4} - \frac{\varphi_1}{2}}{\left[\frac{(1-e\sin\varphi_1)}{(1+e\sin\varphi_1)}\right]^{\frac{1}{2}}}$	$t_2 = \frac{\tan\frac{\pi}{4} - \frac{\varphi_2}{2}}{\left[\frac{(1-e\sin\varphi_2)}{(1+e\sin\varphi_2)}\right]^{\frac{1}{2}}}$

Tabla II.3.2: Formulación para el cálculo de las coordenadas LLA a partir de las coordenadas cartesianas.

$$\boxed{\text{longitud} = \lambda = \frac{\theta}{n} + \lambda_0} \tag{II.3.3}$$

El cálculo de la latitud es un proceso iterativo que es necesario realizar como sigue:

1. Calcular t $\Rightarrow$ $t = \sqrt[n]{\frac{\rho}{aF}}$
2. Calcular $\varphi_{inicial} \Rightarrow \text{longitud} = \varphi = \frac{\pi}{2} - 2\arctan t$
3. Calcular φ_n iterativamente hasta que la diferencia entre la latitud entre dos iteraciones sea menor que un cierto valor. En este caso se ha establecido 0.000000001. $\Rightarrow$

$$\boxed{\text{latitud} = \varphi = \frac{\pi}{2} - 2\arctan t\left[\frac{1-e\sin\varphi}{1+e\sin\varphi}\right]^{\frac{e}{2}}} \tag{II.3.4}$$

Capítulo II.4

El módulo FSUIPC

Resumen

Uno de los objetivos del proyecto es integrar los algoritmos de geo-apuntamiento y geo-localización en una plataforma de simulación. Esta integración permite comprobar de manera visual el correcto funcionamiento de los mismos. La herramienta con la cual se va a realizar la integración, es el programa *Microsoft Flight Simulator*.
Para integrar los algoritmos implementados en el programa *Microsoft Flight Simulator*, es necesario utilizar el módulo FSUIPC (*Flight Simulator Universal Inter-Process Communication*) que permite modificar la visual del programa y ajustarla a nuestras necesidades.
A lo largo del capítulo se hace una breve introducción a la librería, las herramientas de ayuda que proporciona, dónde obtener información adicional y cómo utilizarla en nuestro programa.

El módulo FSUIPC (*Flight Simulator Universal Inter-Process Communication*) [[FSU04a], [FSU04d], [FSU04c] y [FSU04b]] está diseñado para comunicarse desde un programa externo con el programa *Microsoft Flight Simulator* (FS). El objetivo es escribir y leer en/de *offsets* de memoria correspondientes a la posición del avión (latitud, longitud y altura) y la actitud del avión (ángulos de *pitch*, *heading* y *yaw*). El módulo ofrece otras muchas funcionalidades, como realizar ajustes en la meteorología, en el tráfico aéreo...

El módulo se puede utilizar con las versiones FS98, FS2000, FS2002 o FS2004 (este proyecto). Con otras versiones, la compatibilidad puede no ser del 100 %. Su utilización está sujeta a una licencia que ha de adquirirse y en este caso ha sido proporcionada por EADS.

El módulo FSUIPC es una SDK (*Software Development Kit*) desarrollada para utilizar con distintos lenguajes; entre ellos, C++, en el que se han desarrollado los algoritmos de geo-apuntamiento y geo-localización. Actualmente, se ha desarrollado una SDK fácil de utilizar y con muchas herramientas para facilitar la labor del programador. Aun así, hay que saber interpretar los resultados y el orden en que es necesario introducir los datos, para que el FS los interprete correctamente.

II.4.1 FS-Interrogate

FS-Interrogate es una de las herramientas incluidas en el módulo FSUIPC. El objetivo principal es mantener una lista con las variables disponibles, indicando el *offset* de memoria, el tipo, el tamaño, la interpretación, si es de lectura, escritura o ambas... Además nos permite en todo momento conocer el valor que toman las variables en un instante de tiempo determinado.

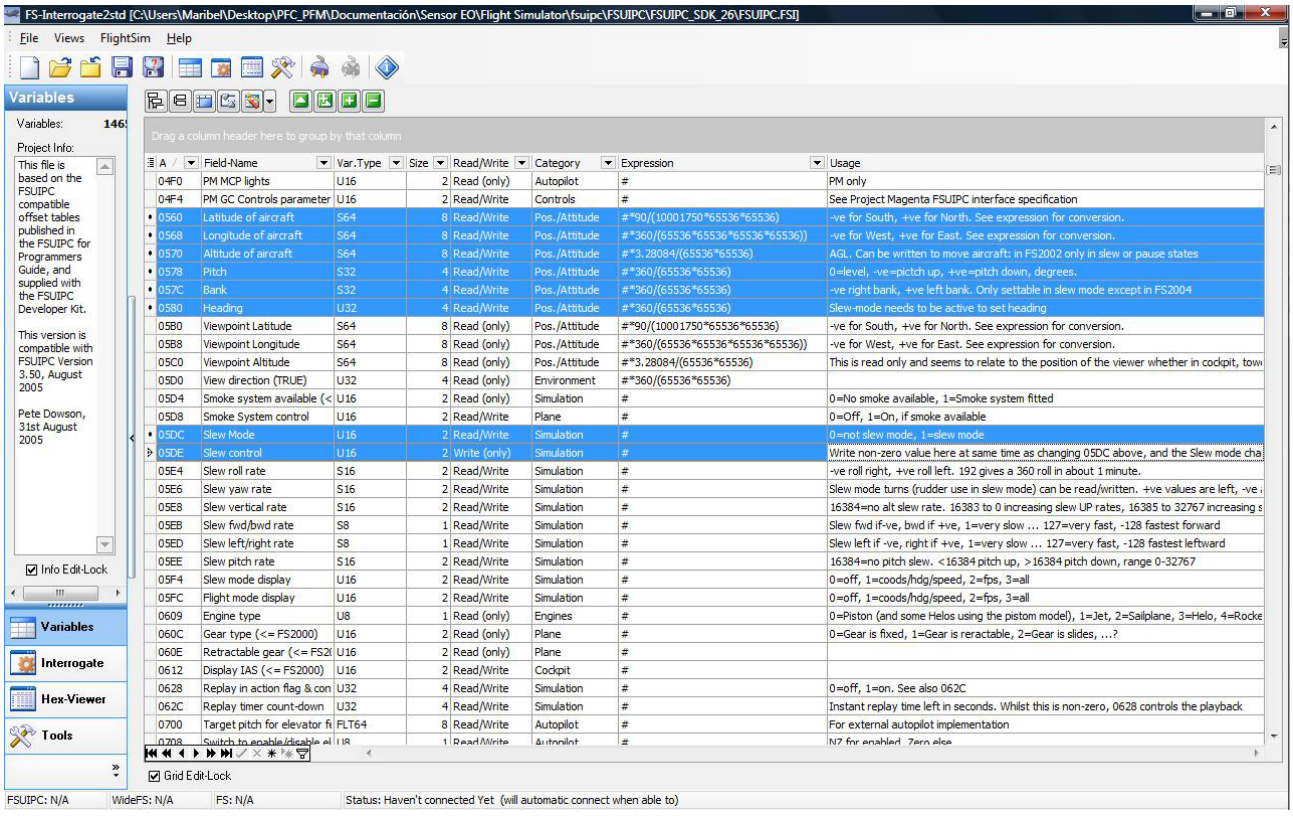


Figura II.4.1: Propiedades de la librería FSUIPC.

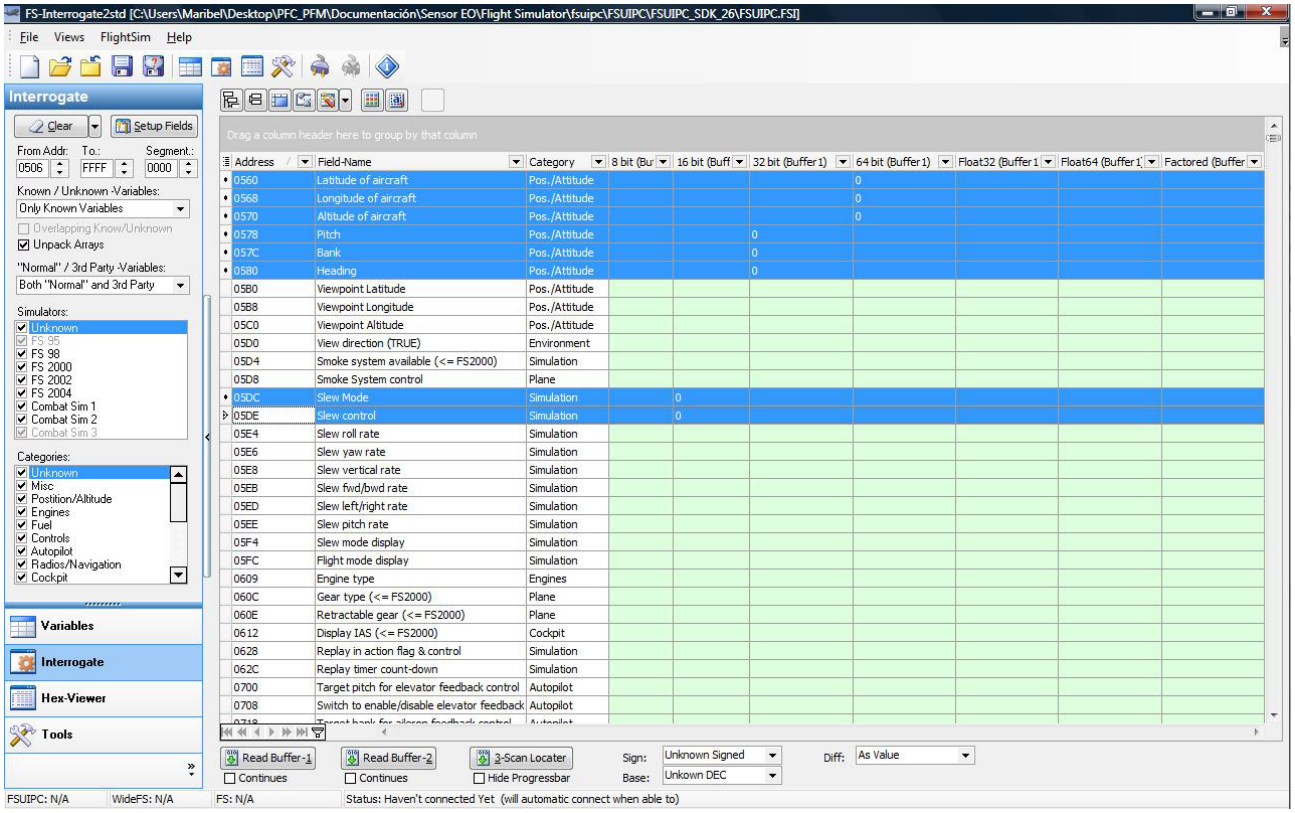


Figura II.4.2: Valores de las propiedades de la librería FSUIPC.

En la figura II.4.1 se muestran las variables y sus propiedades. En azul, están marcadas las que se van a utilizar en este desarrollo. En la figura se indica: la dirección de memoria, el nombre, la longitud, el tipo, si es de escritura, lectura o ambas, la expresión asociada y el modo de utilización.

En la figura II.4.2 se muestran las variables y el valor que toman en un instante determinado. Se puede pedir que esté actualizando continuamente o en el instante que interese. En azul aparecen las variables relevantes para este proyecto.

II.4.2 Utilización del módulo FSUIPC

La librería está diseñada para utilizarse con distintos lenguajes de programación, entre ellos C++, que es el que se está utilizando en la implementación de la aplicación. Para utilizarla es necesario incluir la librería FSUIPC_User.h y tener en el mismo directorio el fichero FSUIPC_User.lib. En la figura II.4.3, se puede ver la estructura del proyecto que se está desarrollando. En rojo, están marcadas las librerías correspondientes al módulo FSUIPC.

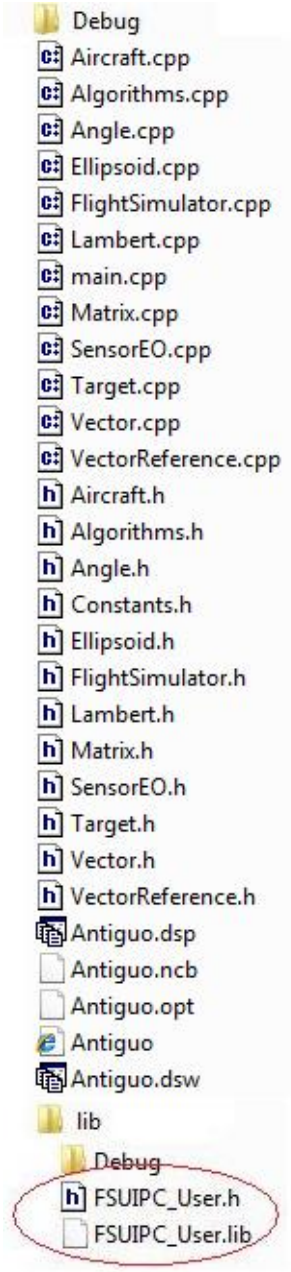


Figura II.4.3: Estructura necesaria para la utilización de FSUIPC.

Es necesario conocer algunos valores importantes de la librería, como los *offset* de memoria

en los cuales se va a escribir o de los cuales se va a leer. Para conocer esto, la librería proporciona un manual [FSU04b] donde se especifican los *offset* de cada una de las variables. También da una breve descripción de la manera correcta de escribir los valores en memoria, para que el FS las interprete correctamente. En la figura II.4.4 se muestran las variables que interesan.

Offset	Size	Use	FS2002	FS2004*
0560	8	Latitude of aircraft in FS units. To convert to Degrees: *If your compiler supports long long (64-bit) integers* then use such a variable to simply copy this 64-bit value into a **double** floating point variable and multiply by 90.0/(10001750.0 * 65536.0 * 65536.0). *Otherwise* you will have to handle the high 32-bits and the low 32-bits separately, combining them into one **double** floating point value (say **dHi**). To do, copy the high part (the 32-bit **int** at 0564) to one **double** and the low part (the 32-bit **unsigned int** at 0560) to another (say **dLo**). Remember that the low part is only *part* of a bigger number, so doesn't have a sign of its own. Divide **dLo** by (65536.0 * 65536.0) to give it its proper magnitude compared to the high part, then either add it to or subtract it from **dHi** according to whether **dHi** is positive or negative. This preserves the integrity of the original positive or negative number. Finally multiply the result by 90.0/10001750.0 to get degrees. Either way, a negative result is South, positive North. [Can be written to move aircraft: in FS2002 only in slew or pause states]	Ok but different	Ok (can set in all modes)
0568	8	Longitude of aircraft in FS format. To convert to Degrees: *If your compiler supports long long (64-bit) integers* then use such a variable to simply copy this 64-bit value into a **double** floating point variable and multiply by 360.0/(65536.0 * 65536.0 * 65536.0 * 65536.0). *Otherwise* you will have to handle the high 32-bits and the low 32-bits separately, combining them into one **double** floating point value (say **dHi**). To do, copy the high part (the 32-bit **int** at 056C) to one **double** and the low part (the 32-bit **unsigned int** at 0568) to another (say **dLo**). Remember that the low part is only *part* of a bigger number, so doesn't have a sign of its own. Divide **dLo** by (65536.0 * 65536.0) to give it its proper magnitude compared to the high part, then either add it to or subtract it from **dHi** according to whether **dHi** is positive or negative. This preserves the integrity of the original positive or negative number. Finally multiply the result by 360.0/(65536.0 * 65536.0) to get degrees. Either way, a negative result is West, positive East. If you did it all unsigned then values over 180.0 represent West longitudes of (360.0 – the value). [Can be written to move aircraft: in FS2002 only in slew or pause states]	Ok but different	Ok (can set in all modes)
0570	8	Altitude, in metres and fractional metres. The units are in the high 32-bit integer (at 0574) and the fractional part is in the low 32-bit integer (at 0570). [Can be written to move aircraft: in FS2002 only in slew or pause states]	Ok but different	Ok (can set in all modes)
0578	4	Pitch, *360/(65536*65536) for degrees. 0=level, –ve=pitch up, +ve=pitch down [Can be set in slew or pause states]	Ok	Ok (can set in all modes)
057C	4	Bank, *360/(65536*65536) for degrees. 0=level, –ve=bank right, +ve=bank left [Can be set in slew or pause states]	Ok	Ok (can set in all modes)
0580	4	Heading, *360/(65536*65536) for degrees TRUE. [Can be set in slew or pause states]	Ok	Ok (can set in all modes)
05DC	2	Slew mode (indicator and control), 0=off, 1=on. (See 05DE also).	Ok	Ok but not like FS2002
05DE	2	Slew control: write non-zero value here *at same time* as changing 05DC above, and the Slew mode change includes the swapping of the assigned joystick axes. [ignored in FS2004 – the axes are swapped in any case. See offset 310B for control of axis connection in slew mode]	Ok	No

Figura II.4.4: Información de las variables del módulo FSUIPC que se van a utilizar.

Es importante introducir correctamente los valores en memoria. En el caso de las variables longitud, latitud, altura, pitch, roll y heading, nos las proporcionan en grados, mientras que el FS entiende *FSUnits*, para lo cual es necesario realizar un cambio de unidades. El cambio de unidades se explica en el documento [FSU04b]. En la tabla II.4.1 se especifican las variables que se van a utilizar en este caso.

latitud (FSUnits)	latitud(°)$*10001750{,}0 * 65536{,}0 * 65536{,}0/90{,}0$
longitud (FSUnits)	longitud(°)$*2^{64}/360{,}0$
altura (FSUnits)	—
pitch (FSUnits)	*pitch*(°)$*2^{32}/360{,}0$
bank (FSUnits)	*bank*(°)$*2^{32}/360{,}0$
heading (FSUnits)	*heading*(°)$*2^{32}/360{,}0$

Tabla II.4.1: Propiedades de FSUIPC utilizadas.

Además del cambio de grados a *FSUnits*, es muy importante tener en cuenta que se están manejando variables de 64 bits, con lo que es necesario manejar variables de tipo __int64 que ofrece el lenguaje C++ [Str00], para realizar la escritura y lectura de las variables.

A continuación, se muestran las líneas de código necesarias para conectar la librería al FS, escribir un valor de latitud en memoria y leer el valor de memoria.

```
1  void connect(){
2          BOOL fsResult;
3          DWORD errorCode;
4
5          fsResult = FSUIPC_Open(SIM_FS2K4, &errorCode);
6          if(fsResult == FALSE){
7                  printError(errorCode);
8          }
9          //Number of licence
10         static char chOurKey[] = "";
11
12         //Registering FSUIPC
13         if (FSUIPC_Write(0x8001,12,chOurKey,&errorCode) == FALSE){
14                 printError(errorCode);
15         }
16    fsResult = FSUIPC_Process(&errorCode);
17         if(fsResult == FALSE){
18                 printError(errorCode);
19         }
20         //Setting Slew mode. Is necesary to modify the memory.
21         short slew = 1;
22         fsResult = FSUIPC_Write(SLEW_MODE, sizeof(short),&slew,&errorCode);
23         if(fsResult == FALSE){
24                 printError(errorCode);
25         }
26         fsResult = FSUIPC_Write(SLEW_CONTROL, sizeof(short),&slew,&errorCode);
27         if(fsResult == FALSE){
28                 printError(errorCode);
29         }
30 }
31 void writeLatitude(double latitude){
32         BOOL fsResult;
33         DWORD errorCode;
34         __int64 lat = (__int64)latitude;
35
36         fsResult = FSUIPC_Write(AC_LATITUDE_LOW, sizeof(__int64),&lat,&errorCode);
37         if(fsResult == FALSE){
38                 printError(errorCode);
39         }
40         fsResult = FSUIPC_Process(&errorCode);
41         if(fsResult == FALSE){
42                         printError(errorCode);
43         }
44 }
45
46 void readLatitude(double latitude){
47         BOOL fsResult;
48         DWORD errorCode;
49         __int64 lat;
50
51         fsResult = FSUIPC_Read(AC_LATITUDE_LOW, sizeof(__int64),&lat,&errorCode);
52         if(fsResult == FALSE){
53                 printFSUIPCError(errorCode);
54         }
55         fsResult = FSUIPC_Process(&errorCode);
56         if(fsResult == FALSE){
57                         printError(errorCode);
58         }
59         printf("Latitude: %f:",lat);
60 }
61 void close(){
62         FSUIPC_Close();
63 }
```

Capítulo II.5

Plan de pruebas

Resumen

Una vez implementados los algoritmos de geo-localización y geo-apuntamiento, es necesario verificar y validar el correcto funcionamiento de los mismos. Para ello, se ha diseñado una batería de pruebas unitarias para realizarlas en PC, y otra batería para la integración con el simulador.

Toda implementación es necesario someterla a una batería de pruebas para cerciorarse del correcto funcionamiento de la misma. Para ello, en esta ocasión se han planteado dos fases, una primera en PC, para eliminar posibles errores de implementación y comprobar de manera analítica el correcto funcionamiento de los algoritmos implementados. Y una segunda fase de integración con una herramienta de simulación para comprobar de manera visual el correcto funcionamiento de los algoritmos. A continuación, se va a desarrollar en que ha consistido cada una de las pruebas en las que consiste la batería de pruebas diseñada.

II.5.1 Pruebas en PC

II.5.1.1 Prueba 1: Cambio de coordenadas LLA $\leftrightarrow$ ECEF

Esta prueba consiste en comprobar que la implementación de cambio de coordenadas LLA a ECEF y viceversa, se realiza de manera correcta. Para ello, se ha determinado hacer uso de la herramienta MATLAB (*MATrix LABoratory*); ya que implementa este cambio de coordenadas, como una función propia de MATLAB [Mat07].

Para realizar las pruebas, se han escogido las coordenadas pertenecientes a tres monumentos importantes: la Torre Eiffel, el Big Ben y el Cristo Redentor del Corcovado.

Para comprobar el funcionamiento, se ejecutan las funciones de cambio de coordenadas en MATLAB y las funciones implementadas en el proyecto. Para que la prueba sea satisfactoria los resultados tienen que coincidir.

Además, es necesario asegurar que para valores extremos no se producen fallos. Las combinaciones probadas son: Latitud -90°, 0° y 90°; Longitud -180°, 0° y 180°, y altura 0 y distinta de cero.

II.5.1.2 Prueba 2: Matrices de cambio de coordenadas

Esta prueba consiste en comprobar que los cambios de coordenadas se están realizando de manera correcta. Se utilizan como base, los resultados obtenidos en el apartado anterior. Las pruebas que se han realizado son las siguientes:

- El producto de cualquier matriz de cambio de coordenadas por su transpuesta ha de ser igual a la matriz identidad. Esto implica:
  1. $C_{NED}^{ECEF} * C_{ECEF}^{NED} = I$
  2. $C_{NED}^{BODY} * C_{BODY}^{NED} = I$
- Si se realiza el proceso completo cíclicamente, el resultado final ha de ser igual al dato de partida.
  1. $\overrightarrow{A_{ECEF}} \rightarrow \overrightarrow{A_{NED}} \rightarrow \overrightarrow{A_{ECEF}}$
     $\overrightarrow{A_{NED}} = C_{ECEF}^{NED} * \overrightarrow{A1_{ECEF}}$
     $\overrightarrow{A2_{ECEF}} = C_{NED}^{ECEF} * \overrightarrow{A_{NED}}$
     Para que la prueba se considere satisfactoria, se tiene que cumplir
     $\overrightarrow{A1_{ECEF}} = \overrightarrow{A2_{ECEF}}$

2. $\overrightarrow{A_{ECEF}} \rightarrow \overrightarrow{A_{NED}} \rightarrow \overrightarrow{A_{BODY}} \rightarrow \overrightarrow{A_{NED}} \rightarrow \overrightarrow{A_{ECEF}}$
   $\overrightarrow{A1_{NED}} = C^{NED}_{ECEF} * \overrightarrow{A1_{ECEF}}$
   $\overrightarrow{A_{BODY}} = C^{BODY}_{NED} * \overrightarrow{A1_{NED}}$
   $\overrightarrow{A2_{NED}} = C^{NED}_{BODY} * \overrightarrow{A_{BODY}}$
   $\overrightarrow{A2_{ECEF}} = C^{ECEF}_{NED} * \overrightarrow{A2_{NED}}$
   Para que la prueba se considere satisfactoria, se tiene que cumplir
   $\overrightarrow{A1_{ECEF}} = \overrightarrow{A2_{ECEF}}$, $\overrightarrow{A1_{NED}} = \overrightarrow{A2_{NED}}$

3. $\overrightarrow{A_{ECEF}} \rightarrow \overrightarrow{A_{BODY}} \rightarrow \overrightarrow{A_{ECEF}}$
   $\overrightarrow{A_{BODY}} = C^{BODY}_{NED} * C^{NED}_{ECEF} * \overrightarrow{A1_{ECEF}}$
   $\overrightarrow{A2_{ECEF}} = C^{ECEF}_{NED} * C^{NED}_{BODY} * \overrightarrow{A2_{NED}}$
   Para que la prueba se considere satisfactoria, se tiene que cumplir
   $\overrightarrow{A1_{ECEF}} = \overrightarrow{A2_{ECEF}}$

II.5.1.3 Prueba 3: Cálculo de la actitud de la cámara

Esta prueba trata de verificar el correcto funcionamiento del cálculo de la actitud de la cámara. Se van a realizar dos pruebas:

- Con los resultados obtenidos en la prueba anterior, ahora se desea comprobar que a partir de los ejes BODY del avión y suponiendo que la cámara se encuentra en el centro de gravedad del mismo, el cálculo de los ángulos azimut y elevación es correcto. Con las mismas premisas se desea comprobar la transformación contraria.

  Para comprobarlo, basta con realizar los siguientes cambios de coordenadas: Coordenadas BODY $\rightarrow$ Azimut-Elevación $\rightarrow$ Coordenadas BODY. Para que la prueba sea satisfactoria, las coordenadas BODY iniciales han de coincidir con las calculadas al final de la cadena.

- Partiendo como en el caso anterior de la premisa de que la cámara está situada en el centro de gravedad, y teniendo en cuenta la característica de que los ejes NED coinciden con los ejes BODY cuando el avión está en vuelo rectilíneo y hacia el norte (ver II.1.4), se va a comprobar que los valores obtenidos de los ángulos de azimut y elevación son correctos.

  Para realizar esta prueba se establece el valor de los ejes X, Y y Z en coordenadas NED; a continuación, se realiza un cambio a coordenadas BODY por medio de la matriz C^{NED}_{BODY}, y se calculan los ángulos de azimut y elevación de la cámara.

  En la tabla se muestran los valores que se han probado y los resultados obtenidos.

Descripción	X_NED_{ECEF}	Y_NED_{ECEF}	Z_NED_{ECEF}	Azimut	Elevación
El TGT está a la misma altura que el AC separado 90m	90°	0°	0°	0°	0°
El TGT está a la misma altura que el AC separado -90m	-90	0	0	180	180
El TGT está situado a 90m en el eje X y a 90m en el eje Y respecto de la posición del AC	90°	90°	0°	45°	0°
El TGT está situado a 90m en el eje X y a 90m en el eje Z respecto de la posición del AC	90°	0°	90°	-45°	0°

Tabla II.5.1: Pruebas del cálculo de la actitud de la cámara

II.5.1.4 Los algoritmos de geo-apuntamiento y geo-localización

Una vez comprobado el funcionamiento de cada una de las partes por separado, es necesario probar todas las partes juntas en el algoritmo.

II.5.1.4.1 Prueba 4: Algoritmo de geo-apuntamiento

Para realizar esta prueba, es necesario realizarla con posiciones conocidas previamente, y así conocer donde debe apuntar la cámara. Se fija la misma longitud y la misma latitud para el TGT y el avión, y se realizan estas pruebas:

- El avión está a una altitud mayor que el TGT. El ángulo de elevación ha de ser 90°.
- El avión está a una altitud inferior que el TGT. El ángulo de elevación ha de ser -90°.

II.5.1.4.2 Prueba 5: Algoritmo de geo-localización

Para comprobar este algoritmo, se introducen los valores obtenidos en el test anterior como entradas al algoritmo de geo-localización. El resultado ha de coincidir con los valores introducidos como entrada en la Prueba 4 o tener un error menor de 1 %. Al igual que en el caso anterior, se realiza el test para los dos casos posibles: que el avión se encuentre posicionado a una altura por encima del objetivo y que se encuentre posicionado a una altura por debajo.

II.5.1.4.3 Prueba 6: Geo-apuntamiento → Geo-localización → Geo-apuntamiento

En esta prueba, se pretende ejecutar los algoritmos cíclicamente utilizando los datos de salida de uno de ellos como entrada del siguiente. Así se comprueba que al final de la cadena, los resultados obtenidos son los mismos que los introducidos al inicio de la misma, o con un error menor de 1 %.

A diferencia de la prueba 5, aquí no es relevante el conocer el resultado de las etapas intermedias. Lo que se quiere probar es si el proceso encadenado se realiza correctamente. Para realizar esta prueba, se escogió como objetivo la Torre Eiffel, cuya latitud es 48.858319° y cuya longitud es 2.294494°.

II.5.1.4.4 Prueba 7: Barrido en todo el rango de posibles valores de las entradas

Esta prueba pretende detectar posibles problemas en los valores límite de los rangos de valores que pueden tomar las distintas variables de entrada. Para comprobarlo, se ejecuta la prueba 5 para todas las posibles combinaciones de datos de entrada, y comprueba que no existía ningún problema. Las variables sobre las que se realiza el barrido son,

- *Pitch*: [-180°,180°]
- *Roll*: [-180°,180°]
- *Heading*: [-180°,180°]
- Longitud: [-180°,180°]
- Latitud: [-90°,90°]
- *Tilt*: [-90°,90°]
- *Panoramic*: [-180°,180°]

Para asegurar que en los extremos de los rangos no existe ningún problema, se realiza un barrido de los ángulos de *tilt* y *panoramic* entre el máximo y el máximo-1°, y entre el mínimo y el mínimo+1°, con saltos de 0.1° para todos los posibles valores. A continuación, se realiza la misma prueba, entre los valores máximo-0.000001° y mínimo+0.000001°.

II.5.2 Pruebas de integración en el banco de Aviónica

Tras comprobar el correcto funcionamiento de los algoritmos en PC, es el momento de integrar el programa desarrollado con el programa Microsoft Flight Simulator; y así comprobar, que los algoritmos funcionan correctamente y la integración se realiza con éxito.

Para realizar estas pruebas se ha accedido al laboratorio de Aviónica del departamento de desarrollo de Software de la división de *Defence & Security* (ver foto II.5.1). En él, EADS dispone de un equipo con el programa Microsoft Flight Simulator 2004, el módulo FSUIPC y los mandos necesarios para manejar el programa, y en su caso el avión.

Figura II.5.1: Banco de pruebas del laboratorio de Aviónica [MIS09].

Para realizar las pruebas de esta sección, se han escogido objetivos reales que aparecen en el programa Microsoft Flight Simulator. Las coordenadas y localización de los mismos pueden verse en la tabla II.5.2 y su situación en la figura II.5.2. Si se observa donde están situados los puntos, se puede ver que se han elegido puntos en cada uno de los cuadrantes en los que se divide la tierra. Esto es así, porque durante el desarrollo se observó que podían existir problemas en los algoritmos dependiendo de la posición LLA en la que estuviese el objetivo. Además, se ha escogido una localización cercana al meridiano de Greenwich como ejemplo de caso extremo. La prueba dio como resultado que no existe tal dependencia.

	Monumento	Latitud	Longitud	Altura
	Torre Eiffel	48.8571566471228°	2.2953919124987°	150 m
	Estatua de la libertad	40.6882013225046°	-74.0453073917532°	55 m
	Coliseo romano	41.8980629596943°	12.4891607312774°	40 m
	Taj Mahal	27.1794452045719°	78.0507440482843°	200 m
	Ópera de Sídney	-33.8567825117146°	151.215199604775°	40 m
	London Eye	51.4991106882527°	-0.118531567585762°	100 m
	Cabo de Hornos	-54.665214588454099°	-65.191952889742453°	1 m

Tabla II.5.2: Objetivos que se han escogido para realizar las pruebas.

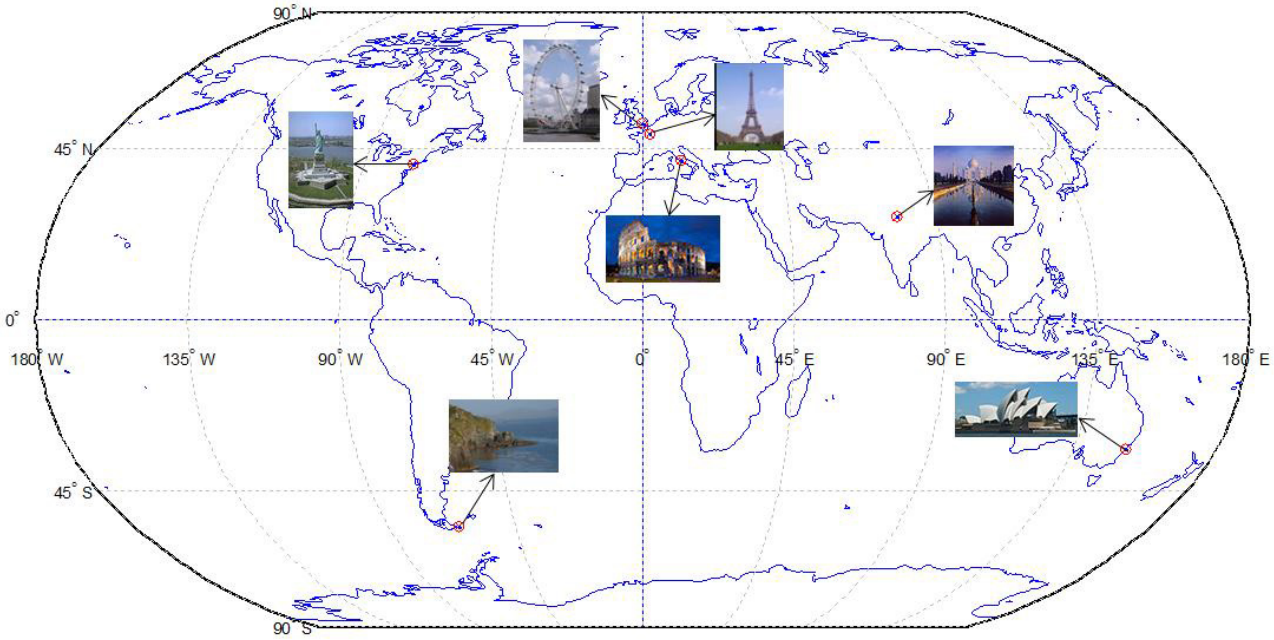


Figura II.5.2: Objetivos que se han escogido para realizar las pruebas.

Es importante destacar que el punto de inicio para todas las simulaciones es el aeropuerto de

la Base aérea de Getafe, que se muestra en la figura II.5.3.

Figura II.5.3: Punto de inicio de las pruebas. Base Aérea de Getafe.

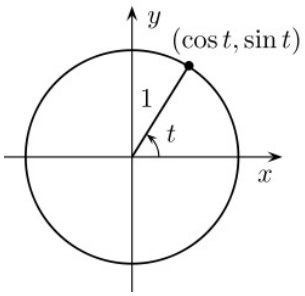
II.5.2.1 Pruebas del algoritmo de geo-apuntamiento

II.5.2.1.1 Prueba 8: Posicionamiento del avión en Torre Eiffel

El objetivo principal de esta prueba es constatar que el acceso a la librería es correcto. Se introducen las coordenadas de la Torre Eiffel como coordenadas del avión y comprueba, cuando se ejecuta el algoritmo, que el avión se posiciona sobre la Torre Eiffel.

II.5.2.1.2 Prueba 9: Apuntamiento a los objetivos realizando círculos alrededor de ellos.

El objetivo es realizar círculos con el avión alrededor de cada uno de los objetivos. Para ello, se plantean las ecuaciones de una circunferencia, por simplicidad se trabaja con la representación en polares, ver figura II.5.4. Una vez obtenidas las coordenadas cartesianas de la posición del avión, es necesario obtener las coordenadas geodésicas o LLA asociadas. Este cambio de coordenadas cartesianas a coordenadas geodésicas o LLA, se realiza mediante la proyección cónica conforme de Lambert, ver II.3. Las coordenadas de la circunferencia se calculan tomando como centro el origen (0,0), ya que la proyección cónica de Lambert se encarga de modificar el centro de la circunferencia y posicionarlo en el TGT.



$x = a + r\cos t \quad y = b + r\sin t$
dado el centro a,b, en nuestro caso $0, 0$
y el radio r de la circunferencia.

Figura II.5.4: Circunferencia en coordenadas polares.

Con los datos de la posición del TGT y del avión, la actitud del avión y la distancia de la cámara al centro de gravedad: se ejecuta el algoritmo de geo-apuntamiento, del cual se obtiene el ángulo de azimut y elevación de la cámara.

Es necesario introducir los valores obtenidos en las variables correspondientes del FS. Para realizarlo, se extrapola el comportamiento de la cámara al comportamiento de la visual del avión. En la tabla II.5.3 se pueden ver las correspondencias que son necesarias realizar.

Latitud	Latitud que se obtiene de transformar la posición (x,y) a coordenadas geodésicas mediante la proyección cónica conforme de Lambert
Longitud	Longitud que se obtiene de transformar la posición (x,y) a coordenadas geodésicas mediante la proyección cónica conforme de Lambert
Ángulo de *Pitch*	Elevación de la cámara obtenida con el algoritmo de geo-apuntamiento
Ángulo de *Heading*	Azimut de la cámara obtenida con el algoritmo de geo-apuntamiento

Tabla II.5.3: Tabla con las extrapolaciones para el FS.

El algoritmo de geo-apuntamiento tiene varios parámetros cuyo valor es posible variar de ejecución en ejecución. Para dar como correcta la prueba es necesario comprobar todas las posibles combinaciones de los parámetros.

A continuación, se listan los parámetros que es posible modificar:

- Distancia entre el avión y el TGT dada por el sensor EO.
- Altura del avión con respecto al TGT.
- Ángulos de *pitch*, *roll* y *heading*.
- Las tres coordenadas de traslación de la cámara con respecto al centro de gravedad del avión.

Las pruebas se realizan para distintos valores y combinaciones de las tres coordenadas de traslación de la cámara con respecto al centro de gravedad del avión; y para distintos ángulos de *pitch*,

roll y *heading*. En el simulador, únicamente se aprecian las diferencias para las combinaciones de los siguientes parámetros,

- La altura del TGT y el avión es la misma.
- La altura del avión es de 0 m
- La altura del avión es de 300 m
- Distancia entre el avión y el objetivo de 0 m.
- Distancia entre el avión y el objetivo de 150 m.
- Distancia entre el avión y el objetivo de 650 m.

Los resultados de esta prueba, como se ha dicho anteriormente son visuales. Se han escogido algunas capturas representativas de los resultados obtenidos y se muestran en la tabla II.5.5.

II.5.2.2 Prueba 10: Algoritmo de geo-localización

Al igual que se realizó para las pruebas en PC, ver II.5.1, es necesario comprobar que los dos algoritmos funcionan. Se ejecuta el algoritmo de geo-apuntamiento, se realiza una lectura de los datos del avión del FS y se ejecuta el algoritmo de geo-localización.

Para admitir la prueba como satisfactoria, al igual que en el caso de las pruebas en PC, la diferencia entre el valor inicial y el valor final tiene que ser menor que un 1 %. Es importante destacar que la prueba se ha realizado para todos las posibles posiciones del avión alrededor de la Torre Eiffel; es decir, para todos los posibles ángulos de la circunferencia y los valores obtenidos han sido siempre los mismos. En la tabla II.5.4, se puede observar que se cumplen las especificaciones y por tanto los algoritmos funcionan correctamente.

	TGT original	TGT Geolocalizacion	Diferencia(%)
Longitud	2,29536	2,28772	0,764 %
Latitud	48,8571	48,8571	0 %
Altura	190	190,001	0,1 %

Tabla II.5.4: Pruebas del algoritmo de geo-localización.

II.5.2.3 Prueba 11: Rendimiento del algoritmo e interacción con el módulo FSUIPC

Esta prueba determina el rendimiento de la aplicación. Durante la fase de comprobación de los algoritmos, se puede ver en el FS que el avión avanzaba dando pequeños saltos en lugar de realizar una trayectoria suave. Antes de realizar esta prueba, se han barajado posibles fuentes y soluciones al problema. A continuación, se detallan cada una de ellas

1. Para calcular la posición del avión en el círculo alrededor de la Torre Eiffel, se hace un barrido de ángulos a lo largo de toda la circunferencia entre 0 grados y 360 grados. La solución propuesta es disminuir el salto del ángulo cuando se realiza el barrido.

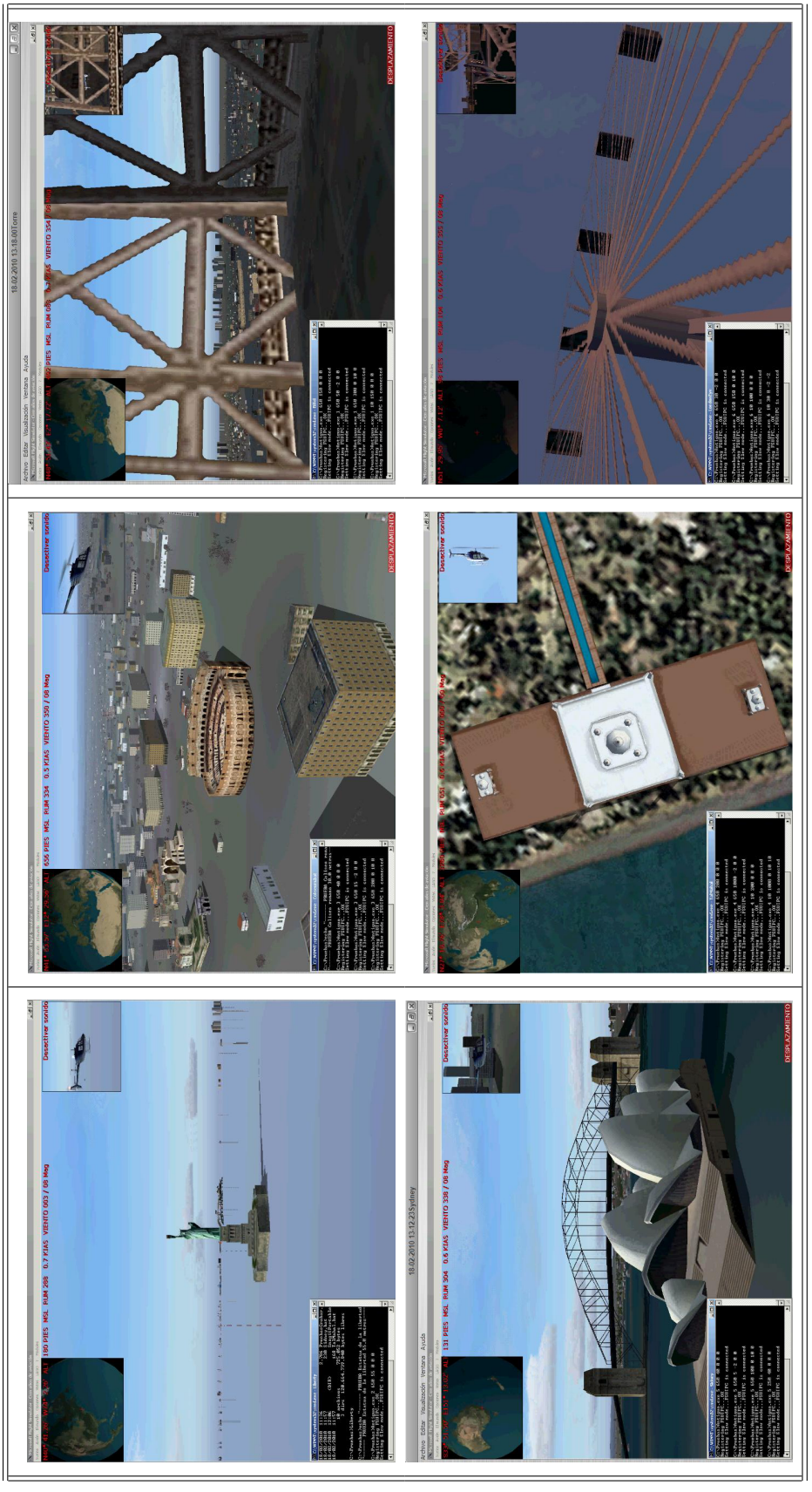


Figura II.5.5: Secuencia de los resultados obtenidos.

2. El algoritmo puede ser demasiado lento, por lo que cabría la posibilidad de modificar la implementación y sustituir las operaciones de coma flotante en operaciones de coma fija.

3. El FS recibe demasiado rápido los datos y no tiene tiempo suficiente para procesarlos y refrescarse con los nuevos datos que llegan. Esto provoca la pérdida de algunos de ellos y por eso se aprecia el efecto de que el avión avanza dando pequeños saltos.

Para averiguar el origen del problema, se ha decidido medir el tiempo de ejecución de las partes importantes de la aplicación: los algoritmos, la interfaz con el módulo FSUIPC, los comandos FSUIPC_Write y process. En la tabla II.5.5 se muestran los resultados obtenidos.

	Valor máximo[µs]	Valor mínimo[µs]
Algoritmo	700	100
Todos los FSUIPC_Write	86	59
Comando Process	26690	48

Tabla II.5.5: Tiempos de ejecución.

Tras analizar los resultados, se ha comprobado que el cuello de botella lo está causando la ejecución del comando process. Este comando tarda mucho más tiempo en ejecutarse que el resto. La primera medida adoptada para solucionar este problema, ha sido disminuir el salto en el barrido. Se ha adoptado esta solución porque es la más sencilla en cuanto a implementación y puede solucionar el problema. Se ha elegido un salto de 0.05° y así, se ha comprobado que se resuelve el problema. Ahora bien, esta solución tiene el inconveniente de que al disminuir el salto, la velocidad de giro disminuye, con lo que la visualización es mucho más lenta. Se ha valorado el hecho de que la visualización sea más lenta frente al hecho de implementar la solución número 3 y se ha decidido admitir la solución como válida.

El computador donde se ha realizado la prueba es un ordenador *Hewlett-Packard*, *Compaq Presario C700 Notebook PC*, con un procesador *Intel Celeron 560@2.13GHz*, con una memoria RAM de 1GB, un sistema operativo de 32 bits y *Windows Vista Home Basic*.

PARTE III

TRATAMIENTO DIGITAL DE IMÁGENES UTILIZANDO SVMs

Durante los siguientes capítulos se va a plantear la fase 2 del proyecto. Tal y como se explicó en la introducción del documento (ver sección I.1.1 y I.1.2), esta fase consiste en la implementación de un clasificador de las imágenes que captura la cámara del UAV. Para ello, en primer lugar se realizará un preprocesado de las imágenes, para a continuación clasificarlas. Una vez implementado el clasificador, se realizará un estudio de viabilidad de la posibilidad de embarcar o no los algoritmos desarrollados.

Como ya se planteó en su momento, las imágenes capturadas por el UAV contienen una gran cantidad de información; por lo que hay que tratarlas para obtener la información relevante y clasificar los posibles objetivos que la formen.

Analicemos la siguiente situación, se han producido unas inundaciones en un pueblo y se manda un UAV para inspeccionar la zona, en un determinado punto se captura una imagen de la situación. Esa imagen es posible analizarla y obtener los posibles objetivos (personas, edificios, fuegos, inundaciones, convoys,...) que contenga y clasificar que tipo de objetivo es cada uno. Esta información ayuda al operador de tierra a determinar la existencia o no de personas y en caso afirmativo, tomar las acciones necesarias para el envío de ayuda para su rescate.

El motivo por el que este proceso se realiza utilizando un computador en lugar de ser realizado por un humano, es que el ojo humano no es capaz de percibir determinados detalles a simple vista. Por esto se ha determinado implementar un clasificador de imágenes como ayuda al operador de tierra a determinar los objetivos existentes.

En la figura 6 se muestra un ejemplo de escenario de clasificación de una imagen.

Para realizar esta clasificación, se va a utilizar un método reconocido y utilizado en muchas aplicaciones debido al potencial que posee. Este método se llama máquina de vectores soporte (SVM). La SVM permite, dada una imagen, clasificar la muestra en la clase a la que pertenece, entre todas las posibles opciones que tenemos en el escenario.

Debido al entorno donde el UAV ha de desarrollar sus misiones, las imágenes que capture van a estar perturbadas por ruido. Además, debido al movimiento del avión sobre la superficie, al realizar las capturas, las imágenes pueden ser capturadas con distintos ángulos; y por tanto, estarán giradas o incluso desplazadas. Es necesario tener en cuenta estos factores durante la implementación del sistema de clasificación. Además se a adoptado como decisión de diseño utilizar imágenes de 16x16 pixels para obtener resultados con mayor facilidad.

Para simular estas propiedades, las imágenes con las que se va a trabajar tendrán las siguientes características:

- Resolución 16x16.
- Rotación tanto en el plano horizontal como en el plano vertical.
- Imágenes sin ruido, con ruido gaussiano y con ruido sal y pimienta.

Para finalizar es necesario precisar algunos puntos importantes en torno a esta fase del proyecto,

- Durante el desarrollo de esta fase, se va a considerar que la imagen que se desea clasificar ha sido aislada de la imagen capturada previamente.
- La cámara que porta el UAV, puede capturar imágenes tanto en el espectro visible como en el infrarrojo. Únicamente se va a tratar con imágenes del espectro visible, aunque el tratamiento es igual, la fase de extracción de características de la muestra diferirá ya que la naturaleza de la imagen no es la misma.

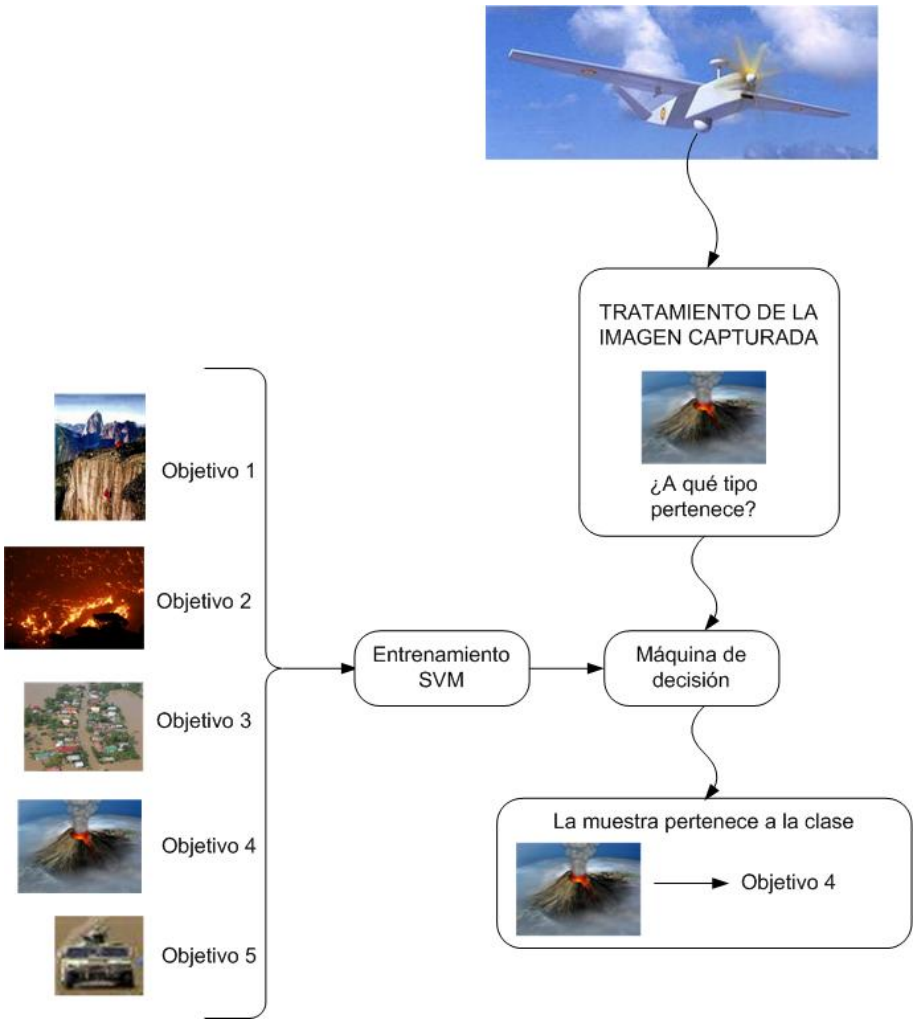


Figura 6: Planteamiento del escenario de clasificación de una imagen.

Los capítulos siguientes presentan el proceso que se ha seguido para la implementación del clasificador de imágenes, la teoría sobre la que se soporta y los resultados obtenidos. Como base del desarrollo de esta parte, se han utilizado investigaciones realizadas con anterioridad y contenidas en [Mar08] y [San].

Capítulo III.1

Extracción de características (PCA)

Resumen

En este capítulo, se explica la técnica de extracción de características utilizada en el proyecto. El objetivo de este tipo de técnicas, es reducir el número de características de las muestras introducidas a la máquina SVM; de esta manera se reduce el coste computacional, se elimina el problema de la dimensionalidad, se aumenta la capacidad de generalización de la máquina SVM y se aporta a la misma la información principal de manera concentrada.

UNa imagen digital se compone por lo general de NxM píxeles, siendo éstos, valores numéricos altos. Estos valores fijan las dimensiones del espacio de entrada del clasificador que deseamos utilizar. La extracción de características nos permite reducir la dimensionalidad de los objetos, manteniendo las características representativas de los mismos y así reducir el coste computacional y el efecto de sobredimensionalidad que afecta a las máquinas de aprendizaje.

La apariencia de un objeto real 3D en una imagen 2D depende de la forma del objeto, de su color, de su posición en la escena global, de sus propiedades reflectivas y de las características de la iluminación de la escena y del sensor empleado para la captación de la misma.

Es importante recordar que en este subproyecto, el objetivo es implementar un clasificador de reconocimiento de imágenes aéreas, con el objetivo de determinar la existencia de blancos enemigos o amigos y estudiar la posibilidad de embarcar estos algoritmos en el avión. Debido a los escenarios donde opera el UAV, es necesario minimizar el tiempo computacional, lo que implica reducir la cantidad de información que manejan los algoritmos.

Las imágenes que recibe como entrada la máquina SVM para realizar el entrenamiento y la evaluación de las muestras, pueden llegar de dos maneras: tratadas, es decir, con las características más relevantes de las mismas, lo que permite reducir la cantidad de información proporcionada a la SVM; o no tratadas (en crudo), lo que provoca un aumento de la información disponible en la máquina, y esto no siempre es recomendable, ya que aumenta el coste computacional, la dimensionalidad del problema y disminuye la capacidad de generalización de la máquina.

Las imágenes con las que se va a trabajar son de dimensiones 16x16 píxeles, y se pueden considerar vectores de 256 componentes (ver figura III.1.1. Los inconvenientes que surgen al tener una dimensión tan elevada, como ya se ha comentado anteriormente, son:

1. **Maldición de la dimensionalidad**: las prestaciones se degradan cuando el número de muestras utilizadas en el entrenamiento es inferior o del mismo orden que la dimensión de las mismas.

2. **Coste computacional**: el tratar con un elevado número de componentes provoca un aumento del coste computacional de manera exponencial.

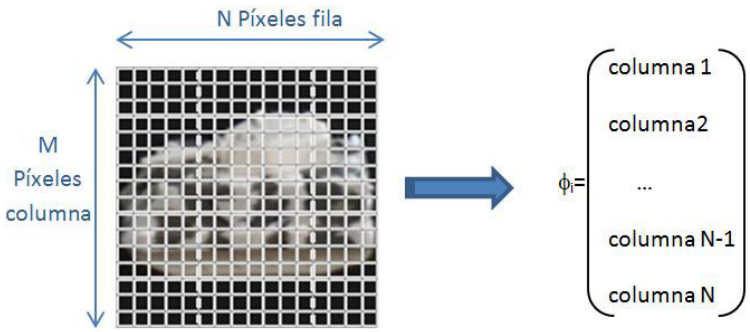


Figura III.1.1: Representación vectorial de una imagen.

El objetivo de las técnicas de extracción de características, es proyectar la imagen en un espacio de dimensión menor que el original, para así reducir el tamaño del vector representativo del objeto. La técnica o el subespacio utilizado con más frecuencia es el creado por los autovectores de la matriz de covarianza de las imágenes de entrenamiento de los objetos, como es el caso de la técnica que se va a estudiar, PCA ([Smi02] y [VOP+02])

El análisis de componentes principales, en inglés, *Principal Component Analysis* (PCA), es una técnica no paramétrica de síntesis de la información, o reducción de la dimensión (número de variables). El objetivo es reducir las dimensiones perdiendo la menor cantidad de información posible.

Esta técnica, fue desarrollada por Pearson en el año 1901 y en 1933 fue Hotelling quien retomó su estudio; pero no fue hasta el boom tecnológico de la informática, cuando se comenzaron a utilizar realmente.

Los nuevos componentes principales o factores, serán una combinación lineal de las variables originales e independientes entre sí. Es un método de identificación de patrones en datos, representando éstos, como una expresión de las similitudes y diferencias de los mismos.

PCA se basa en encontrar las componentes principales que representan a los datos, obteniendo un nuevo espacio de características con la máxima energía (varianza) y mínima correlación. La varianza está relacionada con la información, así, las componentes con máxima varianza aportan la información más importante, y las de mínima varianza corresponden al ruido. Además, cuanto mayor es la correlación entre los datos, menor es el número de características necesarias para conseguir una buena representación de las muestras.

La base de PCA, es proyectar los datos en espacios de características ortogonales, para que el error cuadrático medio entre las observaciones y sus proyecciones, sea mínimo. Cuando el número de características de las imágenes es muy elevado, se tiende a reducir, escogiendo las que posean menor redundancia de información; lo que implica, que sean más incorreladas y por tanto, significativas. Para realizar este proceso, se trasforma el conjunto original en otro conjunto lo más incorrelado posible, realizando así un estudio de la correlación existente entre las características de las muestras. Las componentes principales son este conjunto incorrelado de características que permite reducir el conjunto original de muestras.

El objetivo final, es crear una base ortonormal donde reflejar las imágenes mediante el cálculo de los autovalores y autovectores correspondientes a las imágenes.

El algoritmo PCA se divide en los siguientes pasos:

Paso 1 Para cada una de las imágenes de dimensiones $N = nxn$, hay que crear un vector columna de dimensiones $1xN$.

$$\Rightarrow \overrightarrow{\phi_i} = \begin{pmatrix} \phi_i^1 \\ \phi_i^2 \\ \dots \\ \phi_i^N \end{pmatrix}$$

Paso 2 Normalizar cada vector imagen con respecto a la media de cada uno de ellos.

$$\widetilde{\phi_i} = \overrightarrow{\phi_i} - \overrightarrow{m} \tag{III.1.1}$$

Paso 3 Combinar los vectores columna $\widetilde{\phi_i}$ de cada una de las imágenes y construir una matriz de NxP, donde P es el número de imágenes.

 ➪ $\overrightarrow{\phi_1}$ ➪ $\widetilde{\phi_1}$  ➪ $\overrightarrow{\phi_2}$ ➪ $\widetilde{\phi_2}$  ➪ $\overrightarrow{\phi_2}$ ➪ $\widetilde{\phi_2}$ ...  ➪ $\overrightarrow{\phi_P}$ ➪ $\widetilde{\phi_P}$	$\Phi = \left(\widetilde{\phi^1} \mid \widetilde{\phi^2} \mid \widetilde{\phi^3} \mid \ldots \mid \widetilde{\phi^P} \right)$

Paso 4 Obtener la matriz de covarianzas de la matriz Φ

$$Cov_{xx} = E\{(x - \mu_x)(x - \mu_x)^T\} = \Phi\Phi^T \tag{III.1.2}$$

Paso 5 Calcular los autovalores y autovectores de la matriz de covarianzas. Como mucho, la matriz puede tener P autovectores, que estarán asociados a P autovalores no nulos, siendo siempre P (número de imágenes) menor que N (tamaño de la imagen (nxn)). Los autovalores no nulos de la matriz, proporcionan información sobre los patrones del conjunto de datos; además de formar la base ortonormal donde se proyectarán los vectores originales, y así obtener un conjunto de vectores con una dimensión menor que el conjunto original. Las matriz de autovalores(λ) y autovectores (V), cumplen la propiedad:

$$\lambda V = Cov_{xx} V \tag{III.1.3}$$

Paso 6 El vector de componentes característicos **A** está formado por los N_c autovectores asociados a los autovalores de mayor valor. Donde N_c el número de características que se desean tener en cuenta en el proceso, siempre dentro del rango [1,N], e ignorando las componentes menos significativas, ya que implican una pérdida muy baja de información. El autovector asociado al autovalor de mayor valor es el que posee mayor variación entre las imágenes.

Paso 7 Cada imagen normalizada con respecto a la media, $\widetilde{\phi_i}$, será proyectada en el subespacio PCA creado. Para ello se calcula el producto entre la imagen normalizada y la matriz de componentes característicos.

$$X_{PCA} = A^T \widetilde{\phi_i} \tag{III.1.4}$$

Las columnas de X_{PCA} contienen los datos y las filas correspondientes a cada una de las componentes principales, donde las primeras filas poseen mayor varianza de los datos y por lo tanto un autovalor de mayor valor.

El objetivo de PCA es transformar los datos originales, ya que están expresados en términos de los patrones entre ellos y son las líneas que más describen las relaciones existentes entre los datos. Con esta técnica, los datos finales que se obtienen, contienen únicamente la información relevante. Así la máquina que más tarde los recibirá, usará las características más importantes del conjunto de datos para generalizar y clasificar el resto de muestras con mayor precisión.

Capítulo III.2

Algoritmo *Complex-Log Mapping* (CLM)

Resumen

En este capítulo se explica la transformación de coordenadas logaritmo complejo. Esta transformación, permite obtener una representación de las imágenes origen que es invariante a rotaciones y desplazamientos.

Debido a la naturaleza de las imágenes que se van a tratar, éstas pueden ser capturadas rotadas en cualquiera de los dos planos principales, vertical y horizontal. El método *Complex-Log Mapping* (CLM) permite realizar una transformación de la imagen original y obtener de ella una representación de la imagen invariante a rotaciones y escalados [Kag91]. En el supuesto de este proyecto, únicamente se va a centrar en la propiedad de invarianza a rotaciones del algoritmo CLM.

Este algoritmo consiste en dos fases.

- En la primera fase, se transforma la imagen binaria en coordenadas cartesianas a otra imagen equivalente en coordenadas polares exponenciales.
- En la segunda fase, se calcula el módulo de la transformada de Fourier (FFT, *Fast Fourier Transform*) de la imagen resultante de la etapa anterior.

III.2.1 Logaritmo Complejo

El algoritmo CLM, transforma una imagen binaria en coordenadas cartesianas a otra equivalente en coordenadas polares exponenciales.

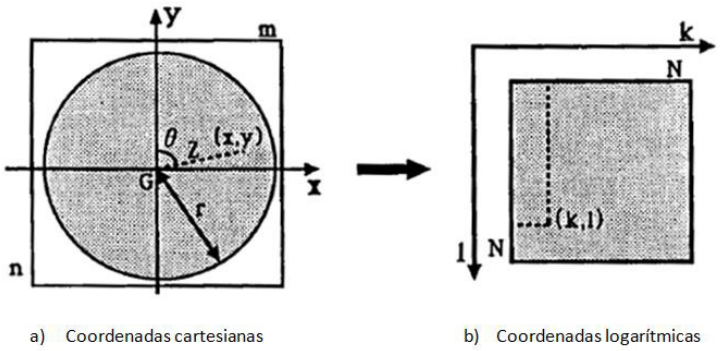


Figura III.2.1: Principio del algoritmo Complex-Log [Kag91].

La figura III.2.1 muestra el principio del algoritmo CLM, el cual consiste en transformar una imagen mxn en una imagen NxN en coordenadas Logaritmo Complejo. En la imagen III.2.1 el pixel (x,y) corresponde al pixel (k,l), siendo la intensidad de la imagen original en el pixel (x,y), la intensidad en el pixel (k,l).

Matemáticamente, se expresa:

$$k = N\frac{\upsilon}{2\pi} \tag{III.2.1}$$

$$l = N \log_r Z \tag{III.2.2}$$

$$\upsilon = \tan^{-1}(\frac{x}{y}) \tag{III.2.3}$$

$$Z = \sqrt{x^2 + y^2} \tag{III.2.4}$$

r es el radio del área elegida para realizar el logaritmo complejo, y se debe elegir tan grande como sea posible para incluir en el proceso la mayor cantidad de información posible de la imagen original.

El algoritmo CLM transforma la rotación y el escalado de las imágenes, en traslaciones horizontales o verticales de la imágenes originales. Este efecto se puede ver en las figuras III.2.2 y III.2.3. En la figura III.2.2 se aplica el algoritmo a dos imágenes rotadas, y se puede comprobar que el resultado es una imagen desplazada horizontalmente en el plano logarítmico complejo. Por el contrario, en la figura III.2.3 el algoritmo se aplica a imágenes escaladas, el resultado es una imagen desplazada verticalmente en el plano logarítmico complejo.

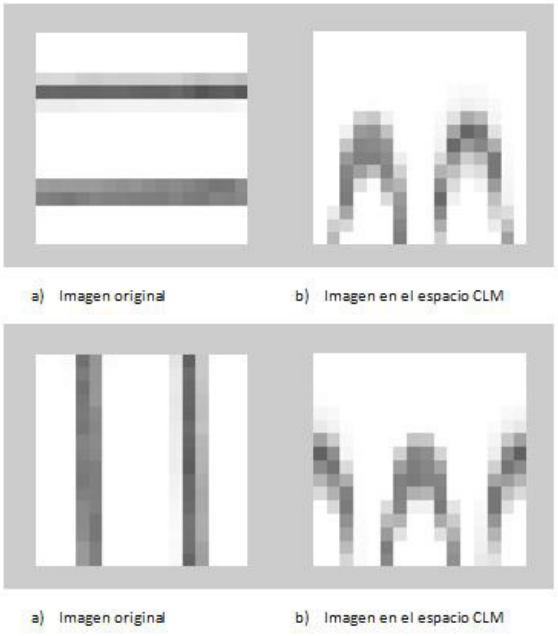


Figura III.2.2: Ejemplo de aplicación del algoritmo CLM a imágenes rotadas.

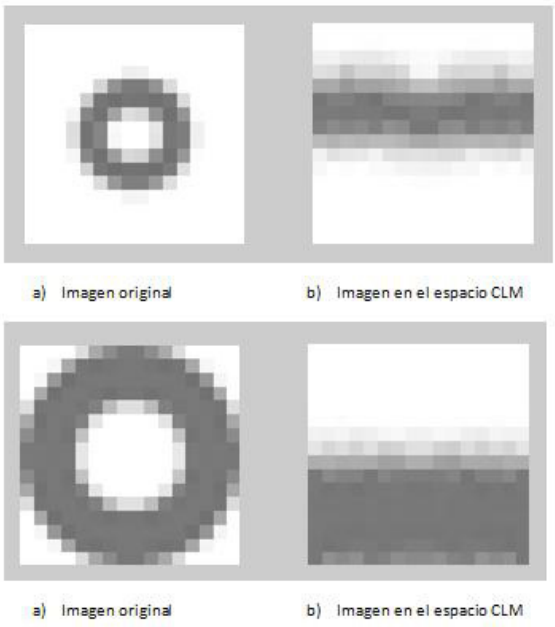


Figura III.2.3: Ejemplo de aplicación del algoritmo CLM a imágenes escaladas.

Es importante destacar la necesidad de que la imagen esté centrada y las rotaciones y cambios en el escalado se hagan con respecto al centro de la imagen, porque, en caso contrario las propiedades del algoritmo se pueden ver alteradas. Este efecto se puede ver en la figura III.2.4.

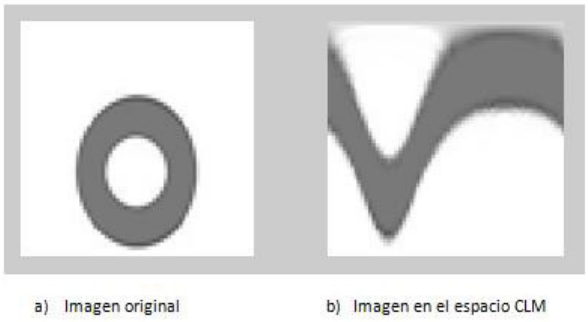


Figura III.2.4: Ejemplo de aplicación del algoritmo CLM a imágenes desplazadas.

III.2.2 Transformada de Fourier (FFT, *Fast Fourier Transform*)

En esta segunda fase, se utiliza la FFT, para tratar adecuadamente cualquier traslación en las imágenes mapeadas en el espacio logarítmico (fase 1) debidas a cambios de rotación y escalados

Dadas dos señales $f_1(t)$ y $f_2(t)$ y cumpliéndose que $f_2(t) = f_1(t-\tau_0)$, se mantiene la siguiente relación entre las Trasformadas de Fourier de f_1 y f_2, $F_1(\omega)$ y $F_2(\omega)$ respectivamente,

$$F_2(\omega) = F_1(\omega)exp(-j\omega\tau_0) \tag{III.2.5}$$

El término $exp(-j\omega\tau_0)$ de la ecuación III.2.5 expresa la componente de fase, con lo que se cumple se cumple,

$$|F_1(\omega)| = |F_2(\omega)|, \tag{III.2.6}$$

donde $|\cdot|$ es el módulo de la componente y por lo tanto el módulo de la componente de la transformada de Fourier es invariante a la traslación τ_0

Capítulo III.3

Clasificadores multiclase

Resumen

En este capítulo se muestran las principales metodologías de clasificadores multiclase. Los resultados del proyecto se van a basar únicamente en la metodología Todos vs Todos (AvA) que es la que implementa la librería LIBSVM.

El número de objetivos entre los cuales se desea clasificar es más de dos, por lo que el clasificador que se ha de implementar ha de ser multiclase [[DD] y [RK04], [ZFM]]; es decir, un clasificador cuyos datos de entrenamiento pertenezcan a N clases diferentes y ante la llegada de una muestra puedan clasificarla dentro de uno de esos N patrones.

Hay que tener en cuenta, que cuanto mayor es el número de clases que intervienen en el entrenamiento, más complicado es realizar la clasificación. Como se ha explicado en la sección I.2.5 la SVM resuelve el problema de clasificación binaria, para el cual es eficiente, precisa y con una alta convergencia; pero cuando el escenario no es binario, surgen los problemas. Por este motivo se han desarrollado los clasificadores multiclase.

Hay dos alternativas para implementar un clasificador multiclase: la formulación multiclase [[CS00], [CS01] y [WW99]] y la implementación basada en clasificadores binarios. Es esta última, la que se va a utilizar en este proyecto.

La implementación basada en clasificadores binarios, consiste en transformar el problema multiclase en N problemas binarios, entrenar N máquinas, clasificar las muestras nuevas y fusionar las predicciones obtenidas de los N clasificadores binarios y así seleccionar una de las N clases como respuesta al problema que se desea resolver.

A continuación se van a exponer los métodos más importantes de los clasificadores multiclase y con los que se van a trabajar en el proyecto.

III.3.1 Uno vs otros - *One vs All* (OvA)

Es el método más simple para solucionar los problemas multiclase. Parte de la suposición de que existen N clases en el problema.

El primer paso es dividir las N clases en dos grupos. El primer grupo contiene todas las muestras pertenecientes a una clase, “el uno”, y las cuales son etiquetadas positivamente. El segundo grupo, llamado “el otro”, contiene el resto de muestras combinadas y son etiquetados negativamente. Este proceso se realiza para cada una de las N clases. A continuación, se realiza el entrenamiento de los N clasificadores binarios obtenidos anteriormente. En el proceso de evaluación, la nueva muestra se evalúa contra los N clasificadores binarios, para así poder determinar si pertenece a una clase o no. Este paso produce N resultados, uno por cada uno de los N clasificadores binarios.

En el caso ideal, sólo un clasificador mostrará un resultado positivo y el resto resultados negativos. En la práctica esto no ocurre así, la mayoría de las veces, una muestra puede ser clasificada positivamente en más de una clase. Esto hace que la función de decisión de la clase a la que pertenece, sea aquella cuyo coste de clasificación sea el argumento que maximice la función de clasificación $C^i = \arg\max_{i=1\ldots k}(w_i x + b_i)$; es decir, aquella cuyo clasificador dé el valor máximo entre todos. Este hecho puede provocar resultados ambiguos, el llamado *falso positivo*.

Uno de los problemas que tiene este método, es que los datos no están balanceados, ya que durante el entrenamiento, el grupo “otros”contendrá muchos más datos que la clase “uno”. Además, el tiempo de entrenamiento de este método es proporcional al número de clases que se tenga el grupo de entrenamiento. Por contra, cada máquina posee toda la información de todas las muestras para la fase de entrenamiento, lo que hace que la precisión del clasificador sea muy alta.

En la figura III.3.1 se ve representado gráficamente lo explicado anteriormente.

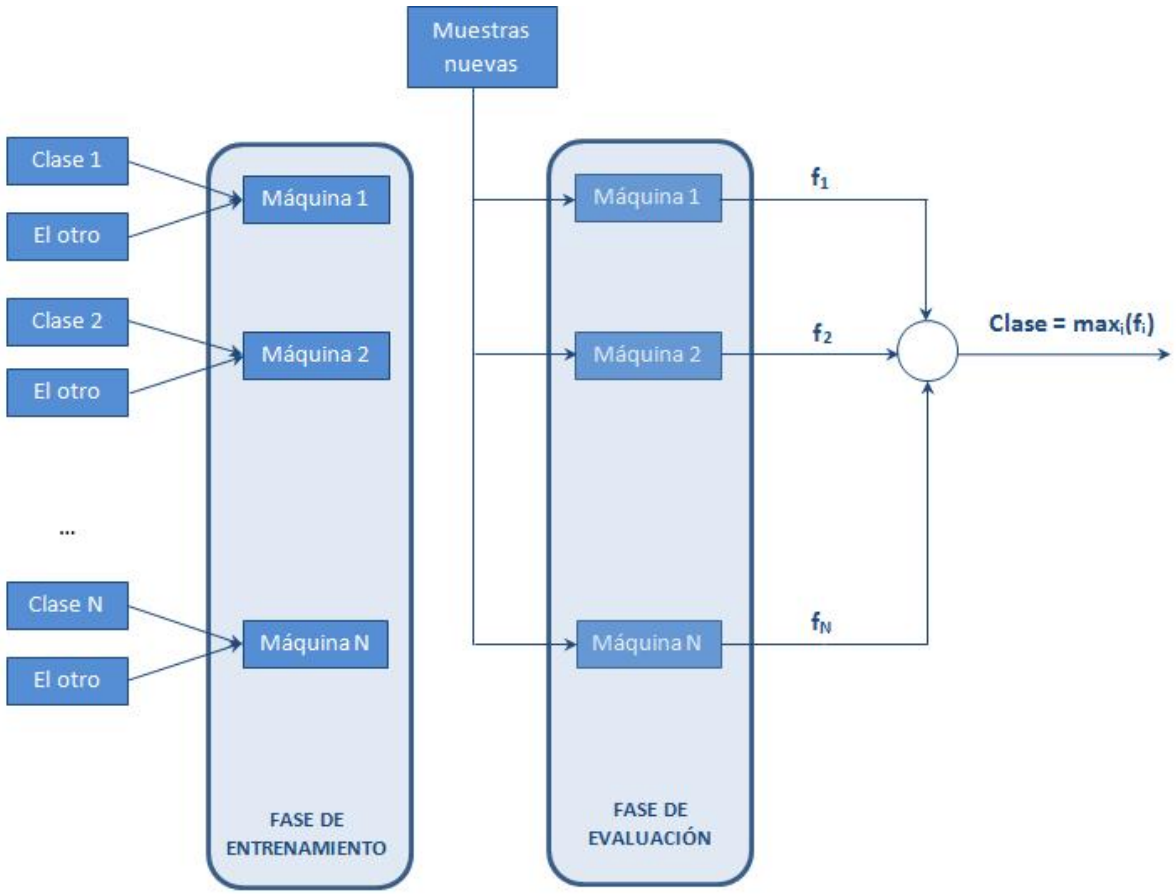


Figura III.3.1: Clasificador multiclase *One vs All*.

Debido al problema de balanceo de los datos, surgen el resto de métodos de clasificación multiclase. Este proyecto se va a centrar en el método Todos contra Todos, ya que es el que implementa la librería LIBSVM. Pero para comprender mejor su funcionamiento es necesario conocer el método Uno único contra Todos.

III.3.2 Uno único vs Todos - *Unique One vs All*

La idea de este método es obtener una predicción no ambigua para un conjunto de muestras dadas y así reducir o eliminar los falsos positivos. El proceso consiste en añadir un segundo paso al método OvA, aplicando clasificaciones discriminatorias binarias entre todas las parejas con predicciones positivas. Para conseguir esto, para cada pareja se entrena el clasificador binario, dando lugar a una predicción positiva (voto) para una de las dos clases. Finalmente, se realiza un recuento de los votos de los clasificadores y la clase con mayor número de votos, representa la decisión final.

El problema del "falso positivo"se elimina en el segundo paso. La decisión final no se toma entre la clase verdadera y los "otros", sino entre dos clases verdaderas. Este método tiene mayor precisión que el anterior. En esencia, la etapa de eliminación del "Falso positivo"es una técnica de reducción de ruido.

En la figura III.3.2 se ve representado gráficamente lo explicado anteriormente.

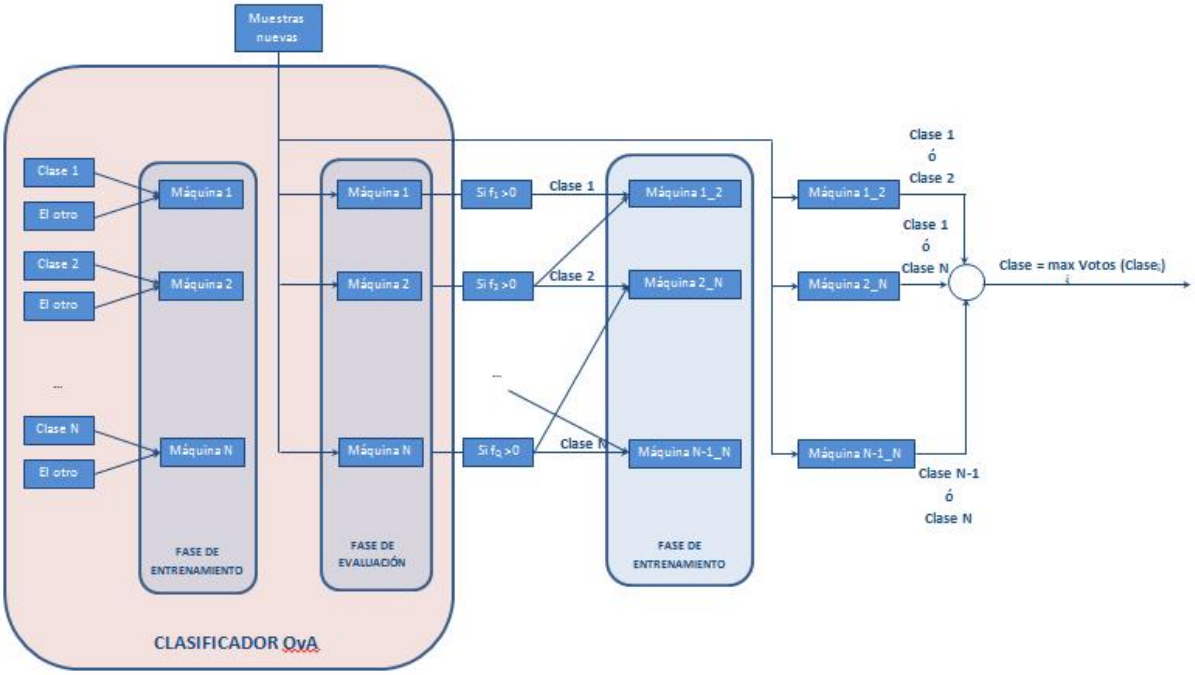


Figura III.3.2: Clasificador multiclase *Unique One vs All*.

III.3.3 Todos vs Todos - *All vs All (AvA)*

En el método "Uno único vs Todos", después de obtener los resultados del método OvA, éstos se introducen en la máquina correspondiente al clasificador binario entre cada una de las dos clases verdaderas obtenidas. Estas máquinas se han entrenado con anterioridad para cada uno de las combinaciones de clasificadores binarios posibles. Así se consiguen eliminar los múltiples positivos. Ahora, se plantea generalizar más y eliminar el método OvA por completo, para ello se ha desarrollado el método "Todos contra Todos"

Este método consiste, en entrenar $N * (N - 1)/2$ clasificadores binarios paralelamente con

dos de las N clases que forman el conjunto de entrenamiento, creando clasificadores para todas las parejas de clases posibles.

Ante la llegada de una muestra nueva, cada clasificador otorga un voto positivo si pertenece a la clase del clasificador. Una vez terminado este paso, se hace un recuento de votos y a la nueva muestra se le asigna la clase con mayor número de votos.

En la realidad, se ha demostrado que el número de votos tiene grandes variaciones y que la clase más popular no tiene porque obtener el mayor número de votos, ya que estos decrecen gradualmente.

Como inconveniente, hay que destacar que la máquina es entrenada sobre un subconjunto que incluye a dos de las clases, por lo que la función de decisión de la máquina, no considera la existencia de otros patrones de entrenamiento pertenecientes a clases distintas. Esto, hay autores, que lo consideran una ventaja, ya que resuelve el problema de balanceo comentado con anterioridad, puesto que la cantidad de datos presentes en cada clase está a la par.

Como desventaja, existe un problema de dimensionalidad, ya que el tamaño de la máquina de clasificación crece de forma cuadrática $(N * (N - 1)/2)$.

En la figura III.3.3 se ve representado gráficamente lo explicado anteriormente.

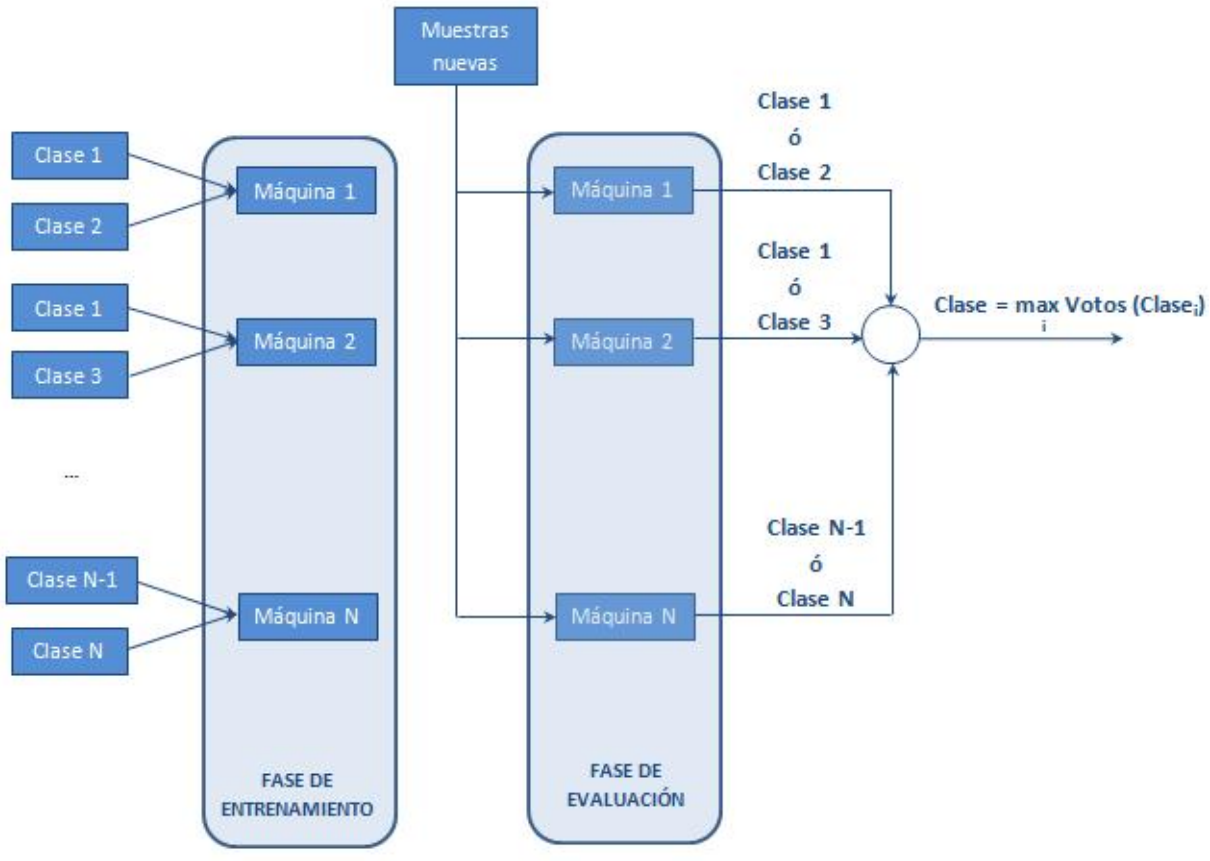


Figura III.3.3: Clasificador multiclase *All vs All*.

Existen discrepancias entre los autores entendidos sobre cual de los métodos es el mejor. En este proyecto se utilizará la metodología AvA o Todos vs Todos, puesto que es la metodología que implementa la librería LIBSVM.

Capítulo III.4

Imágenes empleadas (COIL-100)

Resumen

Debido a la imposibilidad de obtener un conjunto suficientemente amplio y real de imágenes para la realización del estudio de las prestaciones de las técnicas que se están analizando, se ha determinado utilizar las librerías de imágenes proporcionadas por el *Computer Vision Laboratory* de la universidad de Columbia [Uni09].

Como se ha explicado en la introducción del proyecto I.1.2, la cámara del UAV proporciona imágenes (en el espectro normal y el infrarrojo) sobre el escenario al que está apuntando. El objetivo es tratar las imágenes y realizar una clasificación del tipo de objetivo que contiene.

El avión que se ha escogido como referencia está en fase de desarrollo, por lo que la cámara que se va a utilizar no está construida. Esto imposibilita la captura y utilización de imágenes reales. Una de las posibles soluciones planteadas, es utilizar imágenes reales captadas por cámaras similares y trabajar sobre ellas. Debido a la división para la que se está realizando el proyecto dentro de EADS (*Defence & Security*), el acceso a esta información no está permitido.

Para salvar estos problemas y poder realizar el estudio, se han extrapolado las características de las imágenes reales a una base de imágenes disponibles y se ha trabajado sobre ella.

Como se ha comentado, el conjunto de imágenes ha de ser suficientemente amplio y característico de las muestras que se desean clasificar, para así poder realizar un entrenamiento característico y una clasificación lo más real posible.

III.4.1 *Columbia Object Image Library* (COIL)-100

La base de imágenes que se ha utilizado para realizar los procesos de entrenamiento, clasificación y validación, es la librería COIL-100 (*Columbia Object Image Library*). Ha sido creada por el departamento CAVE (*Columbia Automated Vision Enviroment*) de la universidad de Columbia [[Ima09] y [Uni09]], que se dedica a la investigación de sistemas avanzados de visión computerizada.

La librería COIL está formada por 7200 imágenes en color, de 100 objetos reales (72 imágenes por objeto), de apariencia, geometría, color y propiedades de reflexión variados (ver figura III.4.1). La adquisición de las imágenes se ha realizado colocando el objeto sobre un fondo negro y un motor giratorio, para poder realizar capturas de las imágenes a diferentes ángulos de rotación con respecto al punto donde está colocada la cámara.

El sensor utilizado para la captura de las imágenes es un sensor de color, que mejora la discriminación de los efectos de la apariencia que sufren los objetos, según la iluminación y la pose que poseen en un determinado instante. Para realizar una captura lo más real posible es necesario realizar diversas capturas, en distintas condiciones de luminosidad y orientación. Las imágenes que se obtienen pueden utilizarse directamente para el entrenamiento, o someterlas a una fase de preprocesado para adaptarlas a las necesidades del proyecto. En este caso, se va a realizar un preprocesado previo a las imágenes.

La librería tal y como se ha explicado anteriormente, está compuesta de 7200 imágenes en color, con una resolución de 128x128, que pertenecen a 100 objetos distintos. En la imagen III.4.1 se muestran los 100 objetos. De cada objeto se realizan 72 capturas, cada imagen tiene una diferencia en el ángulo de rotación de 5°con respecto a la anterior. La imagen III.4.2 muestra 25 capturas de un mismo objeto.

Figura III.4.1: Banco de imágenes de la librería COIL-100.

Figura III.4.2: Capturas de un mismo objeto de la base de imágenes COIL-100.

III.4.2 Procesado de las imágenes

Debido a la imposibilidad de obtener imágenes aéreas reales, es necesario extrapolar las características de éstas a las imágenes de la base de imágenes que se va a utilizar. Además, también es necesario adaptar las imágenes de la COIL a los algoritmos que se les va a aplicar. Para realizar todas estas trasformaciones, MATLAB (*MATrix LABoratory*) ofrece la *Toolbox* de procesado de imagen [Too09] y que se puede consultar en la ayuda que proporciona MATLAB [Mat07].

El procesado se va a realizar en torno a la forma y la calidad de la imagen.

- Procesado de la forma de la imagen

  1. **Resolución**. La resolución de las imágenes originales es de 128x128. Para que exista un compromiso entre el tiempo computacional y la pérdida de características se ha adoptado una resolución de 16x16.
  2. **Giros**. Las imágenes aéreas están giradas tanto en el plano horizontal como en el plano vertical. El giro en el plano horizontal lo da la propia librería COIL y el giro en el plano vertical es el que se va a generar. Se han realizado giros cada grado a lo largo de 360°.
  3. **Desplazamientos**. Con esta manipulación se desea simular un desplazamiento en la imagen, este efecto no se ha tenido en cuenta en este proyecto.
  4. **Escalados**. Las imágenes aéreas pueden captarse desde distintas alturas, así se simula esta propiedad. Este efecto no se ha tenido en cuenta en este proyecto.
  5. **Intensidad de colores**. Para aumentar la velocidad de procesado, se ha tomado como convención transformar las imágenes de RGB (*Red-Green-Blue*) o escala de colores, en escala de grises. Con esta transformación los pixels tomarán valores entre 0 (negro) y 255 (blanco) siguiendo la siguiente fórmula,

$$E = 0{,}31R + 0{,}59G + 0{,}10B \tag{III.4.1}$$

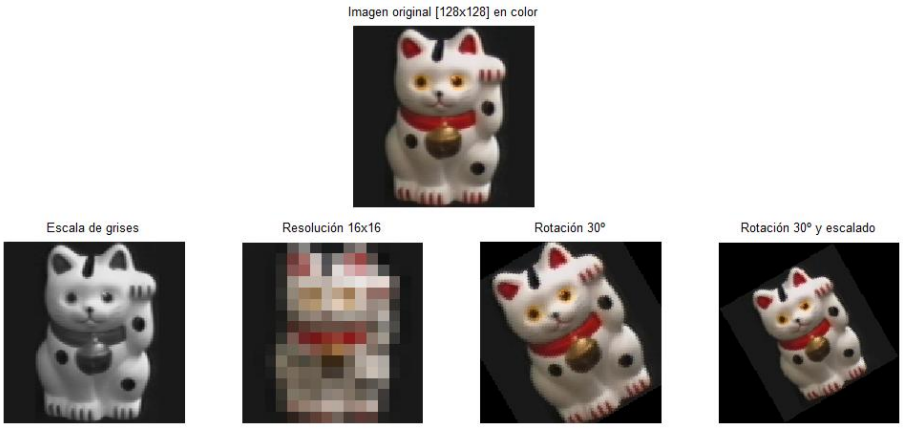


Figura III.4.3: Ejemplo del procesado de una imagen.

- Procesado de la calidad de la imagen
  Debido a los entornos en los que va a operar el avión UAV, la calidad de la imagen puede no ser muy buena. Para simular este efecto, se ha añadido ruido al conjunto de imágenes de la base de datos rotadas de 16x16 pixels, tanto en el conjunto de entrenamiento como en el de test.

Se va a trabajar con dos tipos de ruido: gaussiano y sal y pimienta. Modificando los parámetros de potencia y densidad de ruido, respectivamente, se variará el efecto del ruido sobre las imágenes. Cuanto mayor sean estos valores, la imagen será cada vez más irreconocible.

- **Ruido gaussiano.** El ruido gaussiano es el ruido cuya densidad de probabilidad responde a una distribución normal (o distribución de Gauss). Los parámetros característicos de la distribución de Gauss es la media de la distribución (m) y la varianza (v). En la figura III.4.4 se puede ver un ejemplo. En este tipo de ruido, cuanto mayor es la varianza, el ruido aumenta. En este proyecto, se ha considerado un ruido gaussiano de media y varianza constante para todos los píxeles de las imágenes.
- **Ruido sal y pimienta.** Consiste en píxeles de valor 0 a 255 distribuidos de forma aleatoria por toda la imagen. La intensidad de este ruido viene dada por la densidad (d) de píxeles afectados. Afecta d veces el número de píxeles de la imagen. En la figura III.4.4 puede verse un ejemplo.

En la imagen III.4.4 pueden verse varios ejemplos de lo explicado con anterioridad.

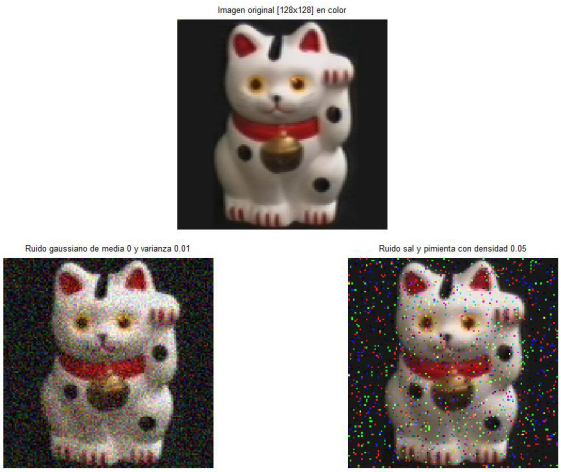


Figura III.4.4: Ejemplo de imagen con ruido gaussiano y sal y pimienta.

Capítulo III.5

Clasificadores de imágenes mediante SVM

Resumen

En este capítulo se va a describir con detalle la arquitectura del clasificador implementado. También se hará una breve reseña de los distintos métodos utilizados y decisiones de diseño que se han tomado durante el desarrollo.

UNa vez explicados los fundamentos teóricos que se van a utilizar en el clasificador de imágenes, se va a explicar cómo se ha implementado, las herramientas y métodos utilizados, las decisiones de diseño que ha sido necesario tomar y la arquitectura final del clasificador.

Como se explicó en la introducción a esta parte del proyecto (ver III), hay dos aspectos a los que se ha de prestar mucha atención,

Baja resolución de las imágenes. Para reducir la cantidad de información disponible en las imágenes, se ha adoptado reducir la resolución de las imágenes a 16x16. La cámara del sensor EO captura imágenes de resolución 1028x1028 píxeles.

Ángulo de captura de las imágenes. El algoritmo CLM, trasforma las imágenes capturadas a una representación invariante a rotaciones y escalados. Esta trasformación permite clasificar correctamente las imágenes, independientemente del ángulo de captura de la imagen.

Extracción de características. Si el número píxeles de las imágenes a tratar resulta muy grande, es interesante reducir la carga y el tiempo computacional, la dimensionalidad del problema y aumentar la generalización de la máquina de clasificación. Para conseguirlo se ha utilizado la técnica de extracción de características PCA.

III.5.1 Librería LIBSVM

La librería LIBSVM proporciona una implementación de las máquinas de vectores soporte, desarrollada por Chih-Chung Chang y Chih-Jen Lin.

El motivo de su elección, es la rapidez en la ejecución de las operaciones, su robustez, el reducido coste computacional y que es de libre distribución. La librería es posible obtenerla en la página web,

http://www.csie.ntu.edu.tw/~cjlin/libsvm/

LIBSVM es un software integrado para clasificación de vectores soporte (C-SVM y nu-SVM), regresión (ϵ-SVR y ν-SVR) y estimación de distribuciones (one-class SVM). Además, la librería soporta clasificación multiclase AvA. LIBSVM es una librería muy extendida y utilizada entre los expertos en este campo.

Las características que ofrece a los usuarios son:

- Facilidad de uso y enlazado con los programas externos.
- Diferentes formulaciones de la SVM.
- Eficiente clasificación multiclase.
- Validación cruzada para la selección del modelo.
- Estimación de probabilidades.
- Implementaciones “poderadas ” (*weighted*) para datos no balanceados.
- Código fuente en C++, Java y C# .NET.
- Implementación en distintos lenguajes, Python, R, MATLAB, Perl, Ruby, Weka e interfaces con CommonL LISP y LabVIEW.

- Distintos tipos de funciones núcleo,
  - Lineal: $K(x_i, x_j) = x_i^T x_j$
  - Polinómico: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$
  - RBF (Radial Basis function): $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \gamma > 0$
  - Sigmoidal: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

  donde γ, r y d son los parámetros de las funciones núcleo.

En este proyecto, se ha decidido utilizar la versión de la librería LIBSVM para MATLAB. Esta decisión está basada en las grandes ventajas que proporciona MATLAB en el manejo de imágenes. Como desventaja de la librería LIBSVM hay que señalar la poca documentación que proporciona. Cuando ha sido necesaria información sobre ella se ha recurrido a internet y a [for10].

La librería LIBSVM implementa el método AvA en su versión de MATLAB como método de clasificación multiclase. Según la FAQ de la librería, AvA y OvA tienen prestaciones muy similares, pero AvA es mucho más simple y el tiempo de entrenamiento es mucho menor. Para hacer estas afirmaciones se han basado en el artículo [HL].

III.5.2 Arquitectura

La implementación se ha dividido en cuatro grandes bloques:

1. Preprocesado de las imágenes.
2. Creación de los conjuntos de entrenamiento y test.
3. Extracción de características.
4. Entrenamiento de la máquina SVM.
5. Test del sistema.

En la figura III.5.1 está detallado el proceso completo y a continuación se van a explicar cada uno de los pasos en detalle.

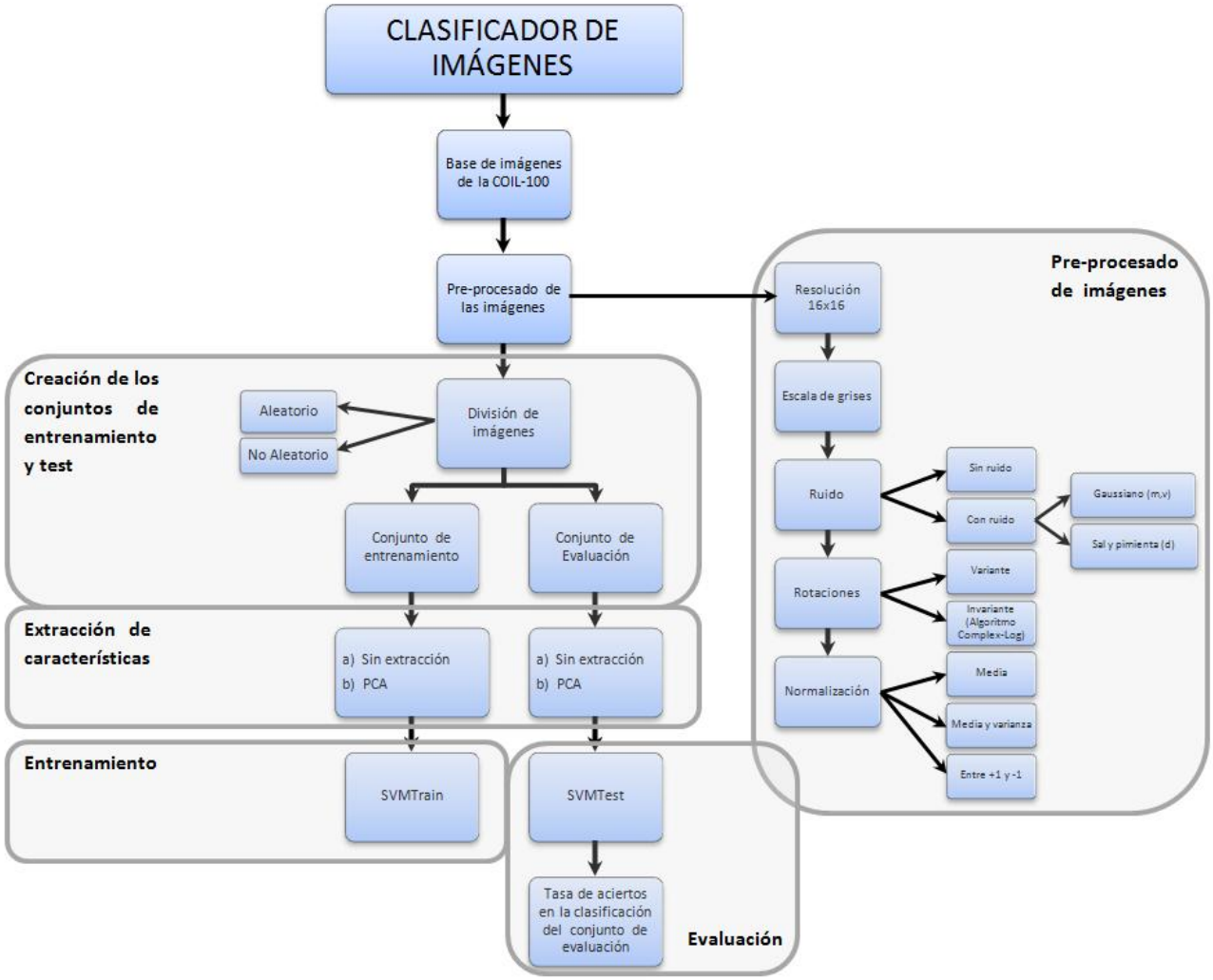


Figura III.5.1: Arquitectura del clasificador de imágenes.

Paso 1 - **Base de imágenes de la COIL-100.** Se adquieren de 1 a N imágenes de la base de imágenes COIL-100 (ver III.4.1), y las etiquetas asociadas a cada una de las imágenes de 1 a N, siendo N el número de clases del clasificador.

Paso 2 - **Preprocesado de las imágenes.** Tal y como se ha explicado en la sección III.4.2, es necesario tratar las imágenes antes de introducirlas en el clasificador. Los tipos de transformaciones que se aplican a las imágenes son:

- Cambio de la resolución a 16x16.
- Cambio de las imágenes de escala de colores a escala de grises.
- Normalización de las imágenes
  - En media.
  - En media y varianza.
  - Entre +1 y -1.
- Aplicación de una función de ruido a las imágenes.
  - Función gaussiana con media m y varianza ν.
  - Función sal y pimienta con densidad d.
- Aplicación del algoritmo Complex-Log para crear una representación de las imágenes invariante a rotaciones y escalados [III.2].

Paso 3 - **Creación de los conjuntos de entrenamiento y test.** Se divide el conjunto de muestras en dos grupos, el de entrenamiento y el de test/evaluación. Esta división se puede realizar de manera aleatoria o no aleatoria. El número de muestras del conjunto de entrenamiento es considerablemente mayor que el del conjunto de evaluación. La relación óptima está entorno a 80 % de muestras de entrenamiento y 20 % de muestras de evaluación.

Paso 4 - **Extracción de características.** Una vez delimitadas cuales son las muestras de entrenamiento, es necesario aplicarles el algoritmo de extracción de características y obtener la matriz de características. Esta matriz se multiplica por los datos de entrenamiento, y el resultado son los datos de entrenamiento de la máquina.

Paso 5 - **Entrenamiento de la máquina SVM.** El entrenamiento de la máquina SVM se realiza utilizando la función SVMTrain de la librería LIBSVM y el conjunto de muestras obtenido con anterioridad. La función de núcleo utilizada es el Kernel Gaussiano, que se caracteriza con dos valores γ y C. Se ha elegido este núcleo porque es con el que obtienen los mejores resultados de generalización.

Paso 6 - **Test del sistema.** Al igual que se realizó con las muestras de entrenamiento, se multiplican las muestras de test por la matriz de características, calculada como resultado de aplicar PCA sobre las muestras de entrenamiento.

Se introducen las muestras de evaluación en la máquina entrenada en el paso anterior. Para ello, se utiliza la función SVMTest de la librería LIBSVM.

Con los resultados obtenidos, se evalúa si la máquina generaliza de manera óptima al clasificar correctamente las muestras no utilizadas en el entrenamiento.

III.5.3 *Complex-Log Mapping*

Esta técnica permite obtener una representación de las imágenes, que hace transparente a la SVM el nivel de rotación y escalado de las mismas. Esta transformación puede ser utilizar como técnica de reducción de características, aunque en este proyecto no se aplica de este modo (ver III.2).

La técnica de CLM que se ha utilizado en este proyecto, compara dos tipos de coordenadas para representar la imagen de NxN píxeles, coordenadas de módulo lineal y coordenadas de módulo logarítmico.

III.5.3.1 Implementación de la transformación *Complex-Log*

En primer lugar, se representan los píxeles que forman la imagen. Para ello, se establecen N puntos en los ejes de abscisas y ordenadas, son los que van a formar la rejilla de píxeles, con un rango de valores entre $[-N+1, N-1]$. En la figura III.5.2, en la imagen superior izquierda, está representada la rejilla con cuadrados azules. Éstos puntos se están representando en coordenadas cartesianas. Los puntos correspondientes en coordenadas polares, están representados en la imagen superior derecha de la figura III.5.2, mediante cuadrados azules. Éstos representan el módulo y la fase correspondiente a cada punto anterior.

Ahora, se desea representar los puntos anteriores en módulo y fase. Para ello, se establece en el eje de abscisas los posibles N módulos con rango $[-N+1, N-1]$, y en el eje de ordenadas las posibles N fases con rango $[-\pi, \pi]$. Los círculos concéntricos al origen de coordenadas que se obtienen como salida, están representados en la figura III.5.2 en la imagen de la derecha, mediante círculos azules. En esta misma imagen, se puede ver que los puntos con igual módulo se encuentran representados en la misma circunferencia, mientras que los puntos con igual fase se encuentran en la misma recta radial.

Si trasformamos los puntos anteriormente descritos de coordenadas cartesianas a coordenadas polares, obtenemos los puntos representados en la imagen superior derecha de la figura III.5.2, mediante cuadrados azules.

En la figura III.5.2 en la parte superior se representa lo explicado anteriormente para un módulo lineal y en la misma figura, pero en la parte inferior, se representa lo mismo pero el módulo es logarítmico; es decir, la distancia entre los valores que tienen la misma fase aumenta según aumenta la distancia al origen.

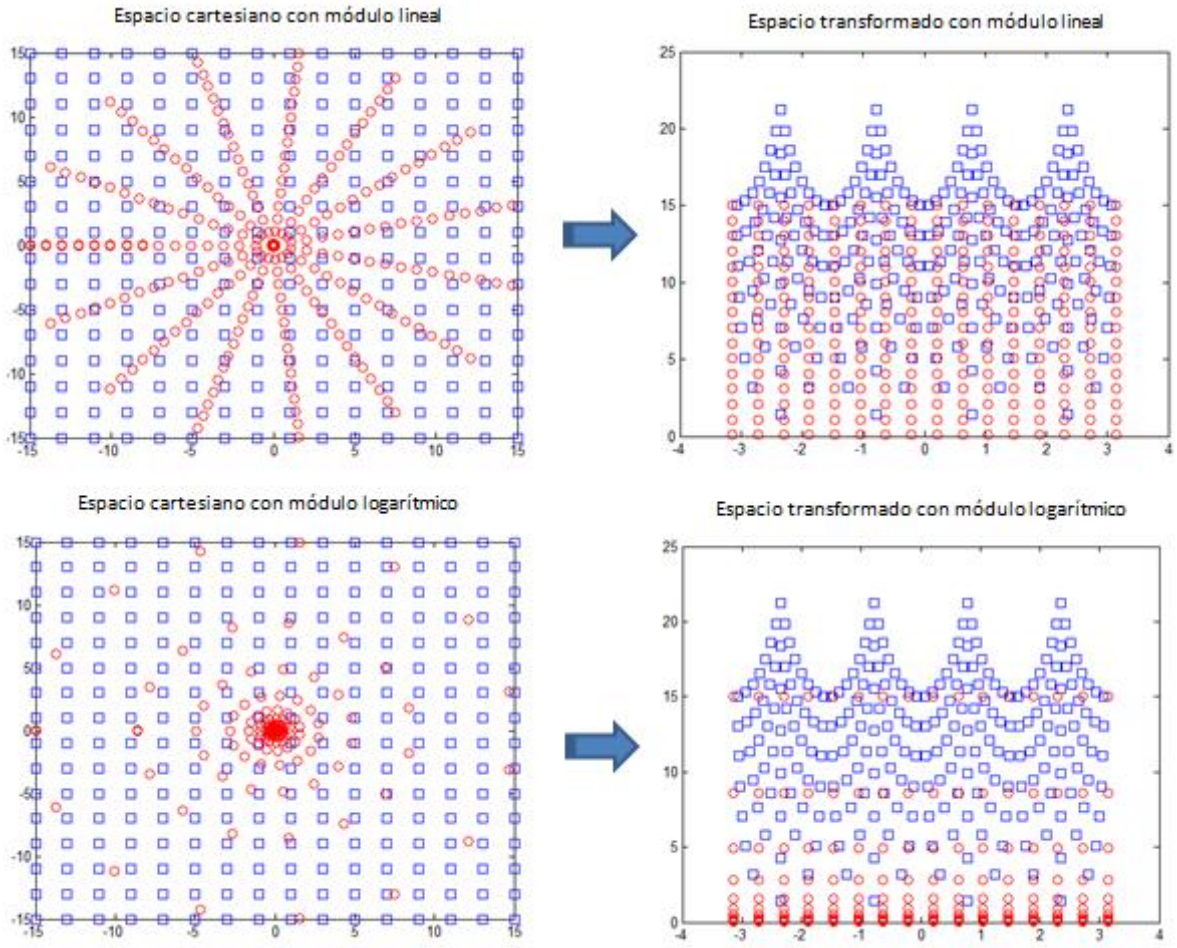


Figura III.5.2: Ejemplos de espacios con módulo lineal y logarítmico.

La representación Logarítmica-Lineal, es de mucha utilidad en imágenes escaladas. Mientras que la transformación de imágenes giradas provoca un desplazamiento en el eje de abscisas (ver III.2.2), la transformación de imágenes escaladas provoca el desplazamiento en el eje de ordenadas (ver figura III.2.3). La representación del módulo Logarítmico permite minimizar este desplazamiento hasta casi eliminarlo. La figura III.5.3 muestra el efecto comentado.

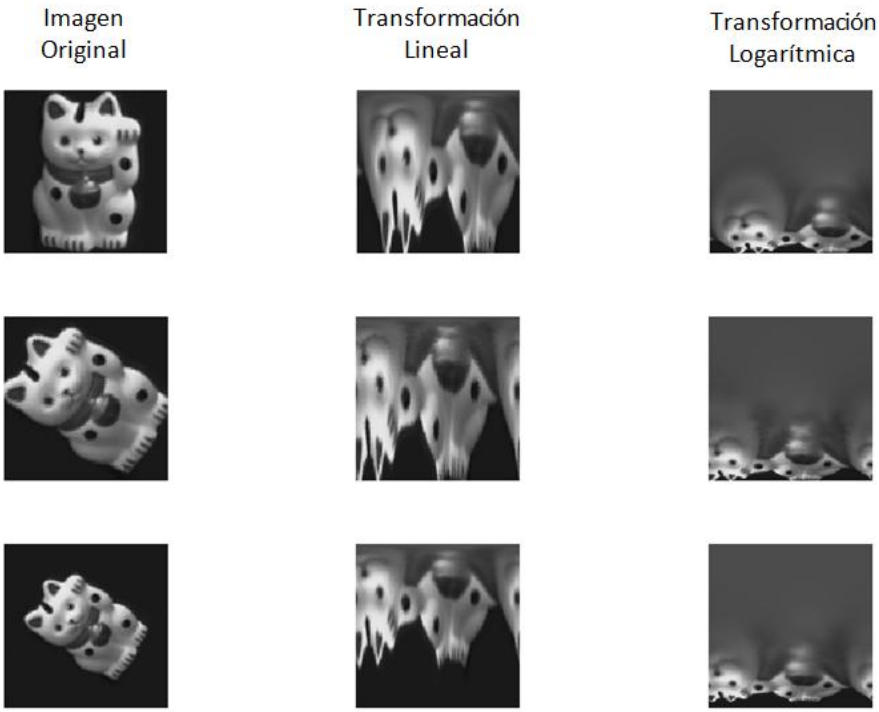


Figura III.5.3: Ejemplo de la trasformación de una imagen en los espacios con módulo lineal y logarítmico.

En este proyecto no se va a trabajar únicamente con imágenes rotadas. Por lo que las ventajas de utilizar un módulo lineal frente a un módulo logarítmico son nulas.

III.5.3.2 Interpolación

La RAE posee diversas acepciones para la palabra interpolar. La acepción matemática es "Calcular el valor aproximado de una magnitud en un intervalo, cuando se conocen algunos de los valores que toma a uno y otro lado de dicho intervalo."

La trasformación de coordenadas cartesianas a coordenadas polares resulta muy sencilla para los ejes. Pero cuando se quiere asignar valores de intensidad a los píxeles en el espacio trasformado, es necesaria la utilización de esta técnica. La interpolación permite encontrar los valores de intensidad en el espacio transformado que corresponden a cada círculo azul de la imagen de la derecha de la figura III.5.3, realizando una ponderación de distancias a los cuadrados más cercanos a su alrededor. Este proceso se realiza con cada valor de intensidad. En la figura III.5.4 se representa un ejemplo de interpolación.

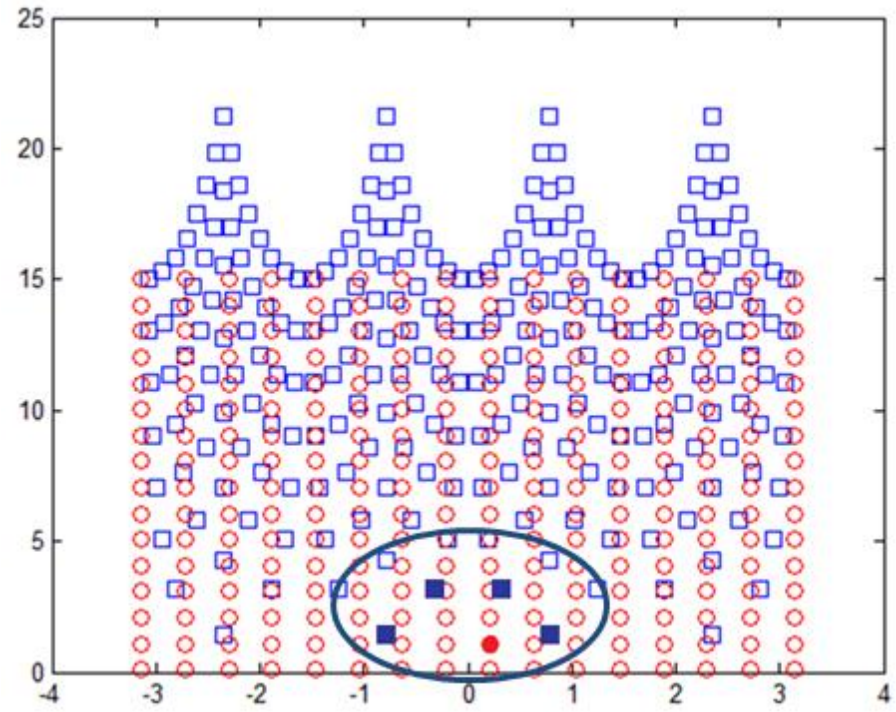


Figura III.5.4: Ejemplo de Interpolación.

Matlab, proporciona distintas funciones de interpolación, en este caso se ha utilizado "interp2". Esta función realiza la interpolación en 2D y proporciona distintos métodos de interpolación:

- "*nearest*": la interpolación se realiza teniendo en cuenta los vecinos más próximos.
- "*linear*": se realiza una interpolación lineal a trozos con los cuatro píxeles que están alrededor. Es la interpolación por defecto y la que se utiliza con mayor frecuencia, ya que es la con la que se obtienen mejores resultados. Es el tipo de interpolación utilizada en este proyecto.
- "*spline*": interpolación *spline* cúbica segmentaria.
- "*cubic*": se realiza con polinomios en dos variables de tercer grado. Requiere que los puntos estén equiespaciados tanto en el eje x, como en el eje y. Si esto no se cumple, es equivalente a la interpolación *spline*.

III.5.4 Extracción de características

Tal y como se explicó en el capítulo III.1, es conveniente realizar una extracción de características de los datos. Esto permite minimizar la dimensionalidad del problema.

En este caso, las imágenes van a tener una resolución de 16x16 píxeles, es decir 256 valores de intensidad. Si el número de imágenes disponibles es alto, la cantidad de información asociada es mucho mayor. La reducción de la cantidad de información que se maneja y el coste computacional son los motivos por los que se utilizan estos métodos.

III.5.5 Elección de la función núcleo y sus parámetros

De todos los núcleos que la LIBSVM proporciona, la función elegida para la implementación desarrollada es RBF. El motivo es ofrece los mejores resultados.

El núcleo RBF tiene como parámetro el valor de γ, que es necesario ajustar durante el desarrollo del clasificador. La figura III.5.5 muestras el principio del núcleo RBF, que consiste en colocar una gaussiana sobre cada muestra con media o y $\gamma = \frac{1}{2v}$, el parámetro a establecer. De esta manera se realiza la clasificación de cada muestra.

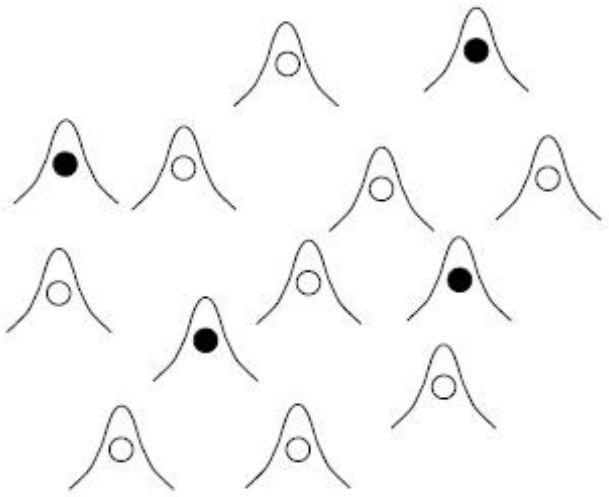


Figura III.5.5: Ejemplo de kernel RBF.

La máquina SVM elegida es C-SVM, que es una SVM en la que se permite definir un parámetro de penalización C, que se impone a las muestras incorrectamente clasificadas.

Estos valores, γ y C, son calculados por medio del procedimiento validación cruzada. Este método consiste en dividir los datos de entrenamiento en dos subconjuntos, el de entrenamiento propiamente dicho y el de validación. Con el primer conjunto se entrena la máquina y con el segundo se calculan los valores óptimos de los parámetros. Para finalizar con la llegada de nuevas muestras, se comprueba la máquina.

El criterio para decidir qué valores de γ y C se han de escoger, ha sido:

> Escoger aquellos valores con los que la probabilidad de acierto del conjunto de validación sea mayor. Si la probabilidad de error es la misma, se escoge el valor que proporcione menor número de vectores soporte.

Capítulo III.6

Análisis de resultados

Resumen

En este capítulo se presentan y analizan los resultados obtenidos tras las diversas simulaciones realizadas para el algoritmo de clasificación de imágenes. Se quieren evaluar las prestaciones que ofrecen las SVMs para el problema que se quiere tratar y las ventajas e inconvenientes de utilizar el algoritmo *Complex-Log* en el proceso de tratamiento de las imágenes utilizadas.

UNa vez realizada la implementación del clasificador se desea comprobar el correcto funcionamiento del mismo, así como obtener resultados para estudiar las prestaciones que ofrece la implementación realizada. Ademas con las pruebas que se han planteado se desea conocer las ventajas e inconvenientes de la utilización del algoritmo *Complex-Log* en el proceso de tratamiento de las imágenes con las que se va a entrenar la máquina y con las que se va a evaluar.

Para realizar las pruebas, se han planteado distintos escenarios que son combinación de las siguientes situaciones:

- Tipo de clasificador que se desea utilizar.
  - Clasificador binario
  - Clasificador multiclase
- Tratamiento de las imágenes utilizadas en entrenamiento y test.
  - Imágenes sin tratar (a partir de ahora, imágenes crudas)
  - Imágenes con ruido
    - Ruido gaussiano
    - Ruido sal y pimienta
  - Imágenes rotadas
    - Sin aplicar el algoritmo *Complex-Log*
    - Aplicando el algoritmo *Complex-Log* con módulo lineal, únicamente fase 1.
- Utilización del algoritmo PCA para la extracción de características de las imágenes.

El primer paso del clasificador es realizar una normalización de los datos, para ello, en el diseño inicial se han planteado tres tipos de normalización: media, media y varianza, y normalización entre +1 y -1. Tras las primeras pruebas, se comprobó que el método de normalización más efectivo para nuestro problema es normalizar en media y varianza todos los datos; por este motivo, todas las simulaciones realizadas cuyos datos se muestran en este capítulo, utilizan esta normalización.

Debido al problema de dimensionalidad que tienen las SVMs, el cual ha sido comentado y explicado a lo largo de la memoria, la resolución de las imágenes ha de ser reducida. Para ello se ha decidido realizar las pruebas con imágenes con una resolución de 16x16 y transformadas a escala de grises; y además aplicar el algoritmo PCA como método de extracción de características que permite también reducir la dimensionalidad del problema. El algoritmo PCA se aplica en todas las pruebas sobre las muestras de entrenamiento. Dependiendo del problema al que se hace frente se han escogido unos valores u otros para el número de características. En cada pruebas se explican detalladamente estos valores.

Las imágenes utilizadas para realizar estas pruebas son las proporcionadas por la librería COIL-100, tal y como se ha venido haciendo a lo largo de la memoria.

En el caso de clasificación binaria, se han escogido dos pares de imágenes de las 100 que ofrece la librería COIL-100 (ver figura III.4.1). Los pares de imágenes escogidas son: dos imágenes muy similares, ver figura III.6.1 y dos imágenes distintas, ver figura III.6.2.

Figura III.6.1: Imágenes de la COIL-100 similares.

Figura III.6.2: Imágenes de la COIL-100 distintas.

Para las pruebas realizadas para la clasificación multiclase, se utilizan 10 imágenes seleccionadas aleatoriamente, de las 100 posibles que ofrece la librería COIL-100 (ver figura III.4.1).

A continuación, se indican otras decisiones de diseño adoptadas para la realización de las pruebas:

- Los elementos de los conjuntos de entrenamiento y test se eligen de manera aleatoria. En el caso de no ejecutar un alto número de veces el clasificador, la elección aleatoria puede provocar alteraciones en los resultados, tal y como se verá en las secciones posteriores.
- Para evitar el efecto de que las imágenes escogidas sean muy parecidas o muy distintas para el proceso de entrenamiento, y puedan alterar los resultados, se ha determinado realizar la clasificación un número elevado de veces y hacer un promedio de los resultados obtenidos. Esto evitará la dependencia que existe de los resultados con las imágenes elegidas durante el proceso de separación en los grupos de entrenamiento y test.
- El número de repeticiones se ha fijado teniendo en cuenta el tiempo computacional que requieren las simulaciones. Los datos elegidos son: para el clasificador binario 100 iteraciones y para el clasificador multiclase 10 iteraciones.

III.6.1 Clasificación de imágenes en crudo

Para las primeras pruebas, se han utilizado imágenes en crudo, es decir, sin añadirles ruido y sin rotarlas, y por lo tanto, sin aplicar el algoritmo *Complex-Log*. Las simulaciones se han realizado para un 50 % y para un 80 % de imágenes en el conjunto de entrenamiento del total de las imágenes utilizadas para esta prueba. En estas primeras pruebas se ha aplicado el algoritmo PCA para todos los posibles valores de número de características (de 1 a 256) dando saltos de 10 características en 10 características.

III.6.1.1 Clasificador binario

En las figuras III.6.3 y III.6.4 se muestran los resultados del clasificador binario.

En las dos imágenes superiores (III.6.3) se muestra la evolución del porcentaje del error que se obtiene al clasificar las muestras de entrenamiento, test y el total de las muestras, para la máquina que se ha obtenido tras el entrenamiento.

Por su parte, las imágenes inferiores (III.6.4) muestran la evolución, para cada número de características, de los siguientes parámetros: número de vectores soporte (SV), C y γ.

Centrando la atención en las imágenes III.6.3, la imagen de la derecha muestra la evolución del error para las dos imágenes similares III.6.1 y la imagen de la izquierda para las dos imágenes distintas III.6.2.

De las dos imágenes III.6.3 se puede deducir:

- Si las imágenes son distintas, el clasificador clasifica siempre correctamente. Cuando no lo son, existe un cierto error, el cual se estabiliza a partir de 50 características.
- Las pruebas se han realizado para que el 50 % y el 80 % de las muestras totales, formen el conjunto de entrenamiento. En la imagen de la derecha se puede apreciar que cuanto mayor es el número de imágenes que forman parte del conjunto de entrenamiento (80 %), mayor es la cantidad de información durante esta fase. Esto provoca que el error de clasificación disminuya considerablemente.
- En estas imágenes se puede observar la evolución de los errores medios de los distintos grupos evaluados: entrenamiento, test y total de muestras. Como es de esperar, el error medio más bajo se consigue con las muestras de entrenamiento, ya que la máquina se ha entrenado utilizando es conjunto. El error medio de test es mayor ya que son muestras nuevas que se introducen para evaluar la capacidad de generalización de la máquina. El error medio del total de las muestras se encuentra siempre entre medias del de entrenamiento y test, ya que el número de errores que se producen para el conjunto de test, se compensa con los producidos para el conjunto de entrenamiento.
- Como se puede observar, cuando las imágenes son similares, el número de errores aumenta frente al que se consigue cuando las muestras son distintas. Esto se debe a que la máquina no es capaz de discernir correctamente cuando las imágenes son muy similares.

En las dos imágenes III.6.4 se puede ver la evolución de los vectores soporte, imagen de la derecha, y la evolución de los parámetros C y γ óptimos, imagen de la izquierda. Estos parámetros son característicos de la fase de entrenamiento de las SVMs. De las imágenes se puede deducir:

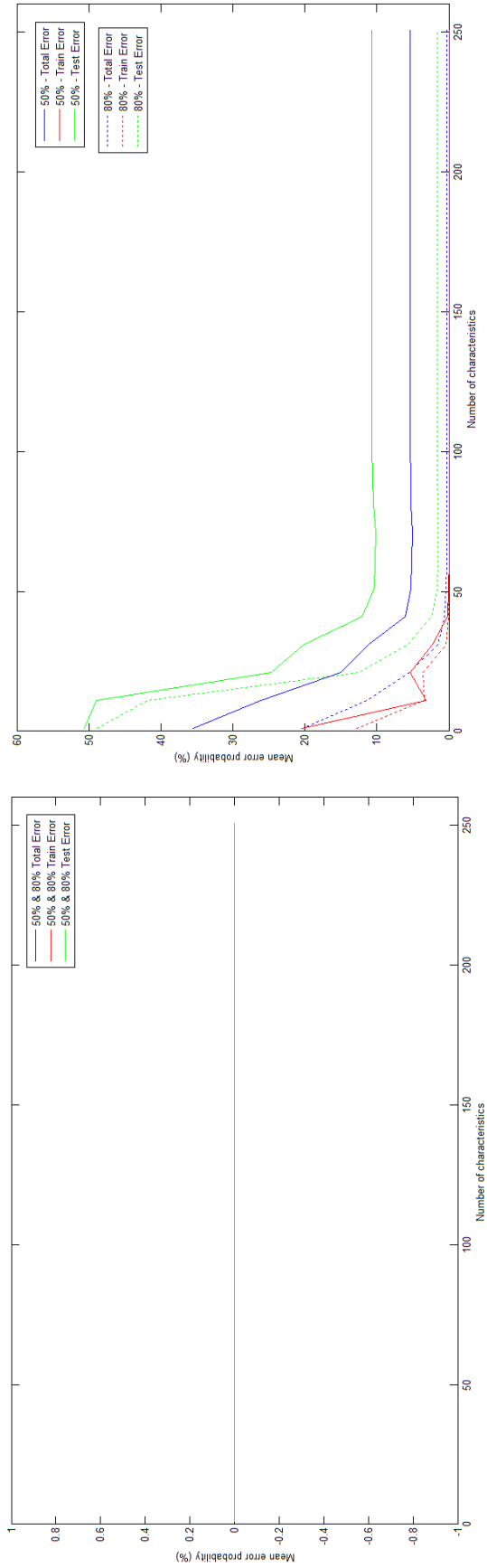


Figura III.6.3: Evolución del % del error para imágenes distintas y similares.

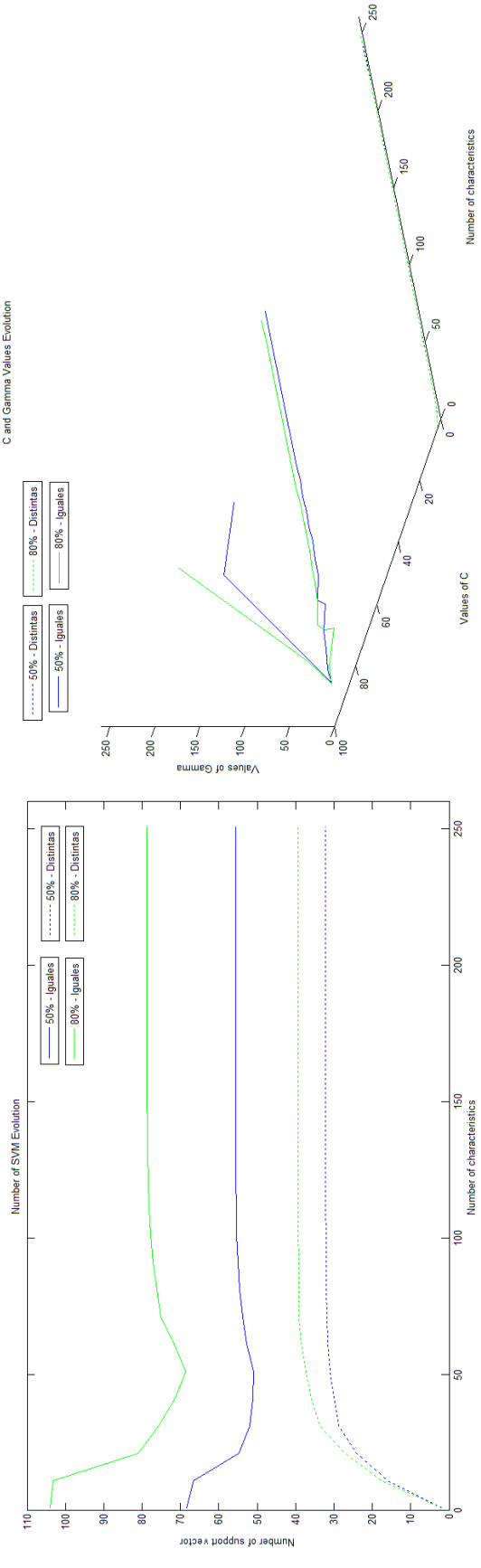


Figura III.6.4: Evolución de los parámetros: número de SV, C y γ.

- Cuanto mayor es la cantidad de información disponible en el entrenamiento de la máquina (80 % vs 50 % de las muestras), mayor es el número de SV resultantes.
- Cuanto mayor es la similitud de las muestras, se requiere un mayor número de SV. Esto se debe a que la máquina necesita más información para poder discernir entre las distintas clases, y así poder generalizar correctamente ante nuevas muestras.
- Cuando las imágenes son similares y el número de características es muy pequeño se necesita un mayor número de vectores soporte. Esto se debe a que es necesario poseer mayor cantidad de información de las imágenes de entrenamiento, y así entrenar la máquina para poder clasificar correctamente las nuevas muestras. El hecho de aumentar el número de características hace que el nivel de información requerida en los vectores soporte disminuya, y por tanto su número también.
- En el caso de imágenes distintas, ocurre lo contrario, el número de SV necesarios para un número de características pequeño es muy bajo, ya que al ser distintas, con 1 característica y el vector adecuado es posible clasificar correctamente las muestras.
- La evolución de los parámetros C y γ queda representada por sus valores óptimos. Estas variables representan el factor de penalización y del factor de varianza del núcleo que se le aplica durante el entrenamiento de la máquina. Se puede ver, que tanto el C como el γ necesario para imágenes distintas es prácticamente nulo, mientras que para las imágenes similares es preciso aumentar los valores. En el caso de imágenes distintas estos valores son muy pequeños ya que la máquina es capaz de discernir con mucha facilidad; en cambio, cuando las imágenes son similares, el valor de C y γ aumenta para conseguir distinguir mejor las muestras.

III.6.1.2 Clasificador multiclase

En el caso del clasificador multiclase, la elección aleatoria de las imágenes va a tener mucha influencia sobre los resultados. Como se ha destacado al inicio del capítulo el número de imágenes escogido es 10. El proceso de entrenamiento de la máquina y clasificación se realiza 10 veces para evitar las posibles dependencias con las imágenes escogidas.

En las figuras III.6.5 y III.6.6 se pueden ver los resultados obtenidos para la clasificación multiclase.

En la imagen III.6.5 se representa la evolución del error para un 50 % y un 80 % de las muestras totales utilizadas para realizar el entrenamiento de la máquina. Las conclusiones que se pueden inferir son muy parecidas a las concluidas para el caso del clasificador binario.

- Cuanto mayor es la cantidad de información utilizada durante el entrenamiento (80 % vs 50 %), menor es el error medio de clasificación que se obtiene.
- A partir de 11 características, el valor medio del error de clasificación se mantiene constante.
- Al igual que en el clasificador binario, el error medio de la clasificación de las muestras de test siempre va a superar al error medio de clasificación de las muestras totales y de entrenamiento.

En la imagen III.6.6 se representa la evolución de los parámetros de entrenamiento número de SV, C y γ. Las conclusiones que se pueden inferir son muy parecidas a las concluidas para el caso del clasificador binario.

- Cuanto mayor es la cantidad de información utilizada durante el entrenamiento (80 % vs 50 %) mayor es el número de SV necesarios.
- El número de SV necesarios con respecto a la clasificación binaria es mayor debido a la cantidad de información necesaria para el entrenamiento correcto de la máquina. Se están utilizando 10 imágenes frente a 2 del caso binario.
- Al igual que en el caso binario, cuanto mayor es la cantidad de información (80 % o número de características), menor es el valor de C y γ ya que la máquina posee más información y es capaz de discernir entre las imágenes.

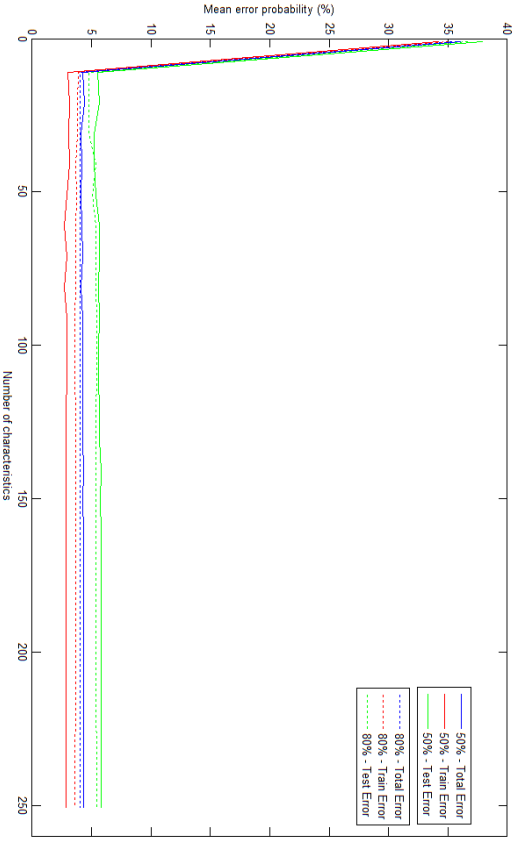


Figura III.6.5: Evolución del % error.

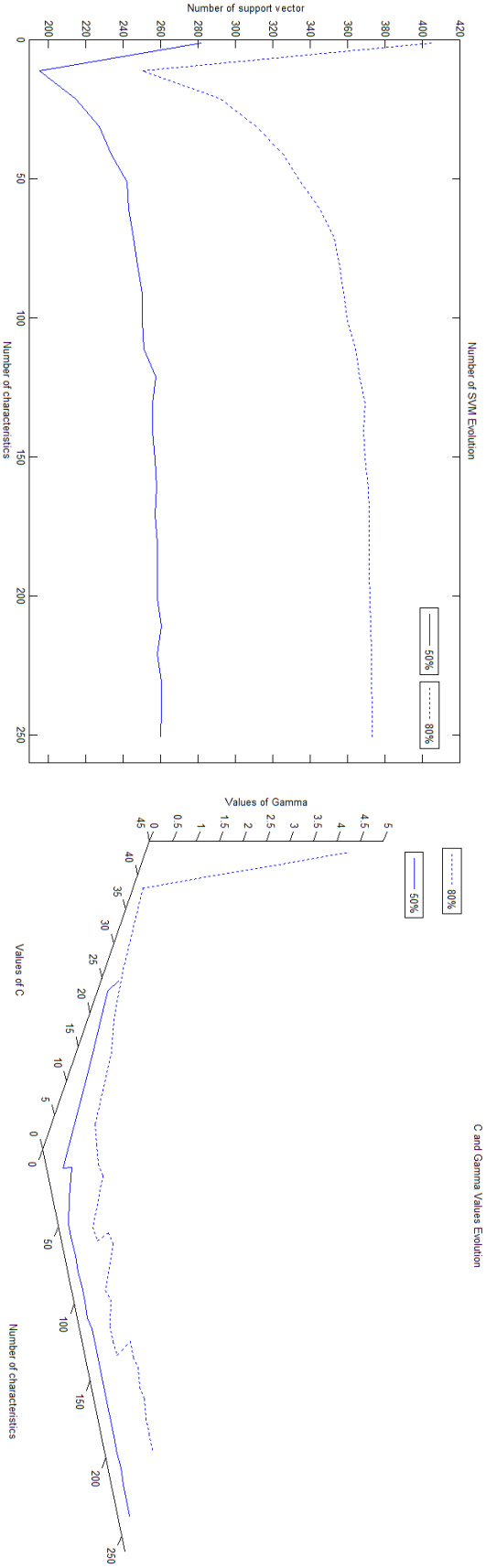


Figura III.6.6: Evolución de los parámetros Número de SV, C y γ.

III.6.2 Clasificación de imágenes con ruido

La imagen III.6.7 muestra los resultados obtenidos para la clasificación de imágenes con ruido. Los tipos de ruido considerados para las pruebas son: gaussiano y sal y pimienta.

Al igual que en las primeras pruebas aquí también se utiliza el algoritmo PCA para realizar extracción de características y así reducir la dimensionalidad del problema. En esta ocasión las pruebas se han centrado en 4 valores específicos de número de características, 11, 51, 101 y 151. Con estos valores se podrá apreciar la evolución las variables a analizar a lo largo de todo el posible conjunto de valores de número de características.

Partiendo de los resultados obtenidos en las secciones III.6.1.1 y III.6.1.2, se concluye:

- En la clasificación binaria de imágenes similares, el error de clasificación es cero a partir de 51 características. Para el caso de imágenes distintas en las pruebas realizadas siempre se ha alcanzado el 0 % de error en la clasificación.
- Cuanto mayor es el ruido de la imagen (alta varianza o densidad de ruido), la imagen se aleja de la imagen real, y por tanto aumenta el error de clasificación de las muestras. Esto produce un aumento del número de SV necesario para obtener un error de entrenamiento lo más bajo posible.
- El número de SV necesarios con respecto a los necesarios en la clasificación de imágenes crudas, es mucho mayor.
- En el caso de la clasificación multiclase, se puede apreciar que el comportamiento debería ser el mismo, pero la influencia de las imágenes elegidas es muy grande con respecto al número de iteraciones realizadas.

Imágenes con ruido			% Train error				% Test error				% Total Error				SV			
			Nchar 11	Nchar 51	Nchar 101	Nchar 251	Nchar 11	Nchar 51	Nchar 101	Nchar 251	Nchar 11	Nchar 51	Nchar 101	Nchar 251	Nchar 11	Nchar 51	Nchar 101	Nchar 251
Clasificación binaria - Imágenes similares	Ruido Gaussiano	v = 0,001	0,1652	0	0	0	5,948	0	0	0	1,3	0	0	0	169	79,17	69,38	103,4
		v = 0,01	0,1913	0	0	0	6,914	0	0	0	1,545	0	0	0	168,2	79,36	95,05	102,4
		v = 0,1	0,2739	0	0	0	7,345	0	0	0	1,698	0	0	0	183,3	81,62	103,6	112,1
		v = 1	0,3522	0	0	0	7,19	0	0	0	1,729	0	0	0	172,3	82,38	103,2	114,7
	Ruido Sal y Pimienta	d = 0,001	0,2	0	0	0	6,431	0	0	0	1,455	0	0	0	167,1	77,07	95,42	99,9
		d = 0,01	0,5565	0	0	0	7,534	0	0	0	1,962	0	0	0	185	81,55	101,9	113,7
		d = 0,1	1,443	0	0	0	17,16	0	0	0	4,608	0	0	0	186,7	107	114,3	137,7
Clasificación binaria - Imágenes distintas	Ruido Gaussiano	v = 0,001	0	0	0	0	0	0	0	0	0	0	0	0	18,99	41,18	42,95	43,02
		v = 0,01	0	0	0	0	0	0	0	0	0	0	0	0	19,24	40,04	42,41	43,83
		v = 0,1	0	0	0	0	0	0	0	0	0	0	0	0	18,31	41,71	49,91	51,88
		v = 1	0	0	0	0	0	0	0	0	0	0	0	0	18,12	52,09	66,57	71,57
	Ruido Sal y Pimienta	d = 0,001	0	0	0	0	0	0	0	0	0	0	0	0	18,59	40,19	42,24	42,9
		d = 0,01	0	0	0	0	0	0	0	0	0	0	0	0	19,06	39,48	40,92	41,7
		d = 0,1	0	0	0	0	0	0	0	0	0	0	0	0	16,54	41,85	49,87	52,96
Clasificación multiclase	Ruido Gaussiano	v = 0,001	3,757	3,652	3,67	3,583	5,034	5,172	5,034	5,103	4,014	3,958	3,944	3,889	307,6	456	507,8	552,2
		v = 0,01	3,757	3,148	3,061	2,87	4,759	4,97	5,103	5,241	3,958	3,514	3,472	3,347	374,2	527	572,8	615,2
		v = 0,1	2,104	1,443	1,496	1,13	2,345	2,414	2,483	2,414	2,153	1,639	1,694	1,389	328	520,2	586,4	653,8
		v = 1	3,443	2,365	2	1,757	5,862	4,828	4,966	4,97	3,931	2,861	2,597	2,40	420	558,8	657,4	717,2
	Ruido Sal y Pimienta	d = 0,001	1,983	1,896	1,896	1,896	2,414	2,414	2,414	2,414	2,069	2	2	2	346	488,6	525,8	552,6
		d = 0,01	7,583	7,53	7,548	7,53	9,793	9,793	9,862	9,862	8,028	7,896	8,014	8	446,2	577,8	611,8	640,2
		d = 0,1	3,791	3,774	3,791	3,739	4,828	4,759	4,759	4,897	4	3,972	3,986	3,972	370	546,8	601.4	624,8

Figura III.6.7: Resultados de la clasificación de imágenes con ruido.

III.6.3 Clasificación de imágenes rotadas

El objetivo de esta tercera fase de pruebas es comprobar la robusted de la fase 1 del algoritmo *Complex-Log*. Se quiere comprobar si con su utilización, es posible prescindir de la presencia de imágenes rotadas en el conjunto de entrenamiento. Estas pruebas vienen propiciadas por la dificultad de conseguir una amplia base de datos de imágenes de los objetivos que se desea clasificar, debido a los entornos en los que opera un UAV.

Para realizar este estudio, se han simulado los siguientes escenarios para los dos pares de imágenes, similares y distintas y para el clasificador multiclase. En todas las pruebas se ha considerado siempre el mismo conjunto de test, así el análisis de resultados se ha podido realizar en igualdad de condiciones para todos los escenarios propuestos.

La librería COIL-100 (III.4.1) proporciona 72 muestras de cada una de las imágenes. Se realiza una rotación a todas las muestras de una misma imagen, el ángulo de rotación aplicado es cada uno de los ángulos comprendidos entre 0° y 360°, en saltos de 20°. El conjunto de test está formado por el 20 % de cada uno de estos grupos de muestras rotadas. Dependiendo del escenario que se esté estudiando, se aplicará a las muestras de cada uno de los conjuntos la fase 1 del algoritmo *Complex-Log* o no.

- Sin aplicar el algoritmo *Complex-Log*

  Escenario 1 . El conjunto de muestras de entrenamiento va ha estar formado por el 80 % de las 72 posibles muestras de cada imagen sin rotar.

  Escenario 2 . El conjunto de muestras de entrenamiento va ha estar formado por el 80 % de las 72 posibles muestras de cada imagen rotadas los ángulos comprendidos entre 0° y 360° en saltos de 40°, 80°, 160° y 320° .

- Aplicando la fase 1 del algoritmo *Complex-Log*

  Escenario 3 . El conjunto de muestras de entrenamiento va ha estar formado por el 80 % de las 72 posibles muestras de cada imagen sin rotar.

  Escenario 4 . El conjunto de muestras de entrenamiento va ha estar formado por el 80 % de las 72 posibles muestras de cada imagen rotadas los angulos comprendidos entre 0° y 360° en saltos de 40°, 80°, 160° y 320° .

A priori, parece razonable, que el Escenario 1 sea el peor de todos, debido a que en el conjunto de entrenamiento no es muy representativo de las muestras de test. En el caso del Escenario 2 según aumenten los saltos de los ángulos utilizados para formar el conjunto de entrenamiento, se tenderá al Escenario 1 y por tanto los resultados empeorarán considerablemente.

En el caso de los escenarios 3 y 4, los resultados deberían mejorar. El hecho de utilizar la fase 1 del algoritmo *Complex-Log*, hace que tanto las muestras de entrenamiento como las de test, sean una representación invariante a rotaciones. Esto debe permitir que aunque el conjunto de entrenamiento no posea muestras rotadas (caso propuesto en el escenario 3), debido a la utilización de la fase 1 del algoritmo *Complex-Log*, la máquina ha de ser capaz de clasificar las muestras de test correctamente.

A continuación, se muestran los resultados obtenidos para cada uno de los escenarios propuestos.

III.6.3.1 Clasificador binario [*Complex-Log* sin FFT (Fase 1)]

En esta sección se van a exponer los resultados obtenidos para el clasificador binario y los 4 escenarios propuestos. Para estas pruebas se han utilizado las imágenes III.6.1 y III.6.2, para contemplar el caso de que las imágenes fuesen muy similares y el caso en el que las imágenes fuesen distintas.

En las figuras III.6.8, III.6.9 y III.6.10, se puede ver la evolución de los errores de entrenamiento y test para cada uno de los escenarios analizados. Todos estos resultados, como ya se ha comentado, han sido obtenidos utilizando un clasificador binario. Es importante resaltar, que para este tipo de clasificadores la peor clasificación se obtiene cuando el error es de un 50 %.

En la imagen III.6.8, se comparan los casos en los que las imágenes de entrenamiento no están giradas y las de test sí. Mientras en el Escenario 1 no se utiliza la fase 1 del algoritmo de *Complex-Log*, en el Escenario 3 sí. Como se puede observar la clasificación de imágenes distintas es mucho mejor que la clasificación de imágenes similares. Los resultados obtenidos utilizando la fase 1 del algoritmo *Complex-Log* y sin utilizarlo son muy parecidos, dependiendo del número de caracteristicas, hay ciertos valores en los que la clasificación es mejor y otros en los que no.

En la figura III.6.11, se puede ver que el número de vectores soporte utilizado cuando se aplica a las imágenes la fase 1 del algoritmo *Complex-Log* es menor que cuando no se utiliza. Esto proporciona mayor rapidez al algoritmo de clasificación.

Los resultados del Escenario 1 eran los esperados, ya que al no tener ninguna referencia en el conjunto de entrenamiento sobre las imágenes rotadas, la clasificación del conjunto de test no podía ser muy buena. En el caso del Escenario 3 se esperaba que con la utilización del algoritmo III.2 mejorara la clasificación y por tanto no fuese necesaria la presencia de las muestras rotadas en el conjunto de entrenamiento.

Sobre los resultados obtenidos para los escenarios 2 y 4, se sigue observando que el hecho de que las imágenes sean diferentes o similares afecta mucho en la clasificación, siendo mucho mejores los resultados obtenidos cuando las imágenes son distintas. Además en este caso si se comparan las imágenes III.6.9 y III.6.10 se puede ver que cuando el número de muestras giradas aumenta en el conjunto de entrenamiento (saltos de 40°), el error de clasificación disminuye. En cambio en el caso de que el conjunto de entrenamiento posea menos muestras giradas (saltos de 160°), la clasificación empeora tanto para imágenes similares como para imágenes diferentes y se aplique la fase 1 del algoritmo *Complex-Log* o no.

En cuanto a los vectores soporte (SV), destacan el aumento de SV en el caso de imágenes similares y realizando saltos de 40°de rotación en las muestras de entrenamiento. En el resto de los casos no se aprecian diferencias notables entre los distintos escenarios propuestos.

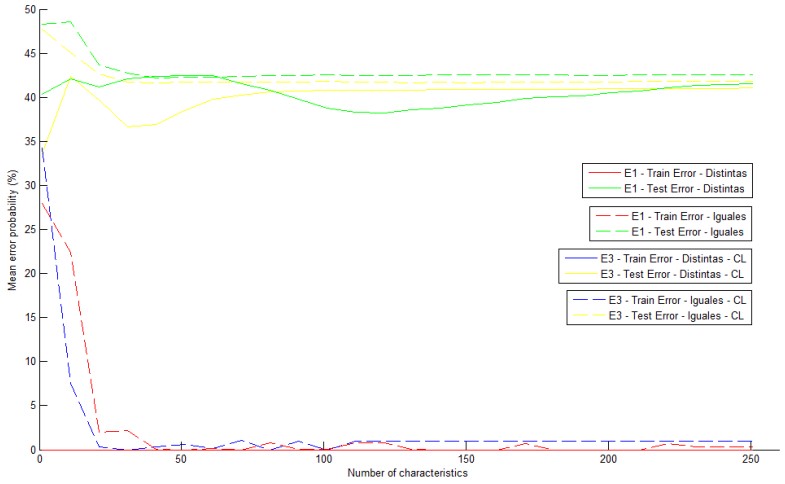


Figura III.6.8: Resultados obtenidos en los escenarios 1 y 3.

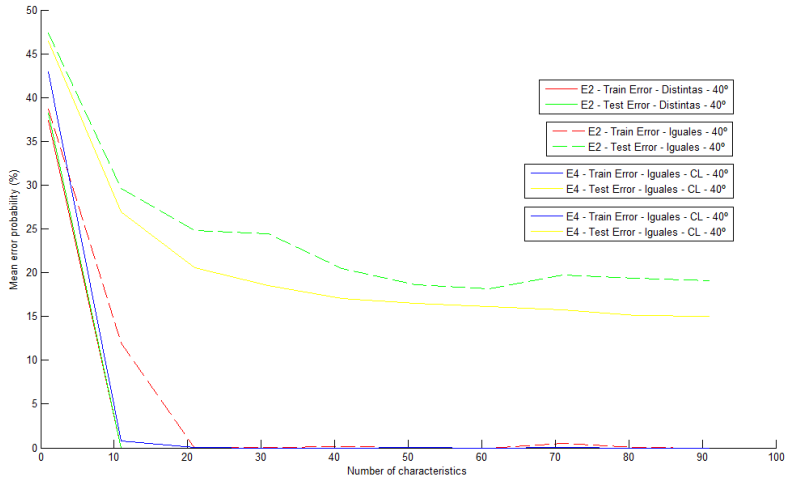


Figura III.6.9: Resultados obtenidos en los escenarios 2 y 4 con saltos de 40° de rotación.

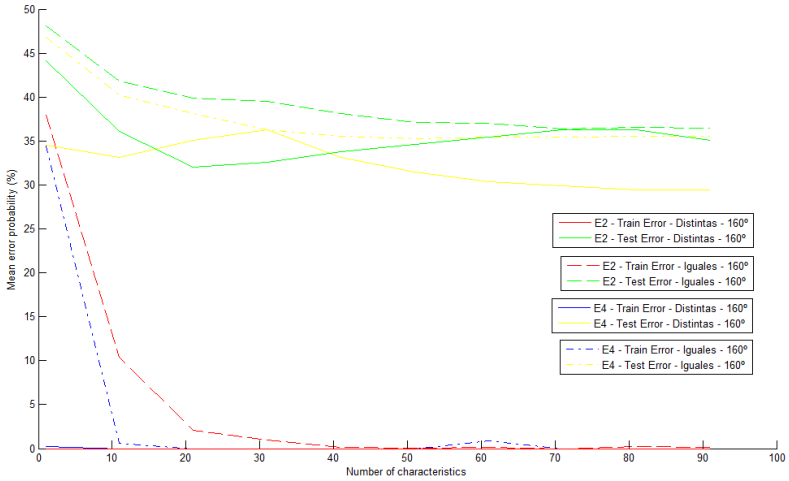


Figura III.6.10: Resultados obtenidos en los escenarios 2 y 4 con saltos de 160° de rotación.

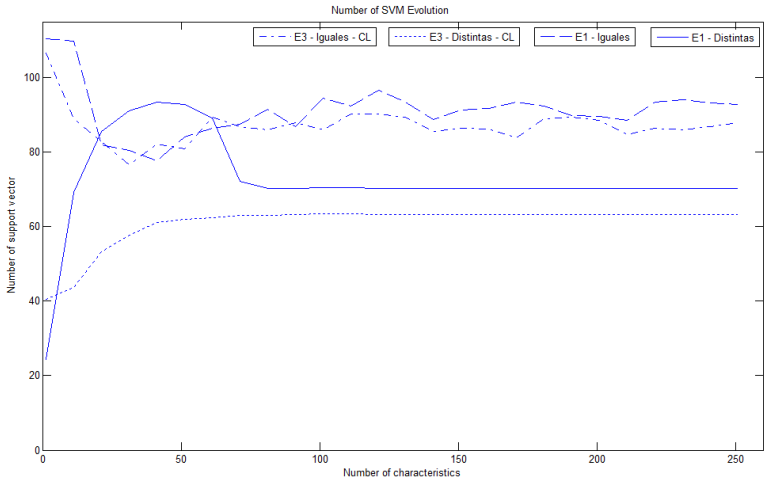


Figura III.6.11: Evolución de los SV en los escenarios 1 y 3.

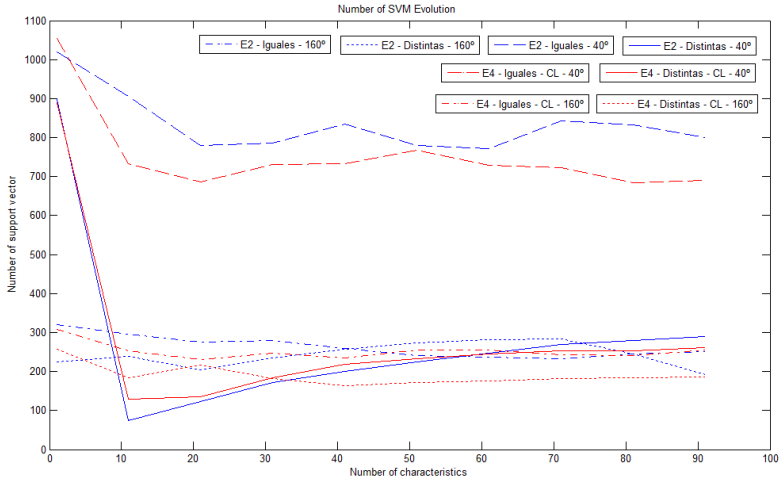


Figura III.6.12: Evolución de los SV en los escenarios 2 y 4 con saltos de 40° y 160° de rotación.

Tras el análisis realizado, se ha podido comprobar que no se han obtenido los resultados esperados. La aplicación de la fase 1 del algoritmo III.2 no mejora las prestaciones de la máquina de clasificación. Una vez conseguidos estos resultados, y analizando el algoritmo implementado, se puede inferir que la segunda fase del mismo, podría mejorar las prestaciones de la máquina. Si se observa la imagen III.5.3, la rotación de la imagen produce una representación desplazada frente a la representación de la imagen sin girar. La segunda fase del algoritmo *Complex-Log* consiste en realizar la FFT, la cual corrige este desplazamiento, además del producido por el escalado de las imágenes. Dada la naturaleza de las imágenes en este problema parece razonable pensar que la inclusión de la fase 2 ayudaría a la máquina a clasificar con mayor precisión.

III.6.3.2 Clasificador binario [*Complex-Log* con FFT]

Tras los resultados obtenidos anteriormente, se ha determinado aplicar las dos fases del algoritmo *Complex-Log* y comprobar si los resultados obtenidos se ajustan a los resultados esperados. Para estas pruebas se han considerado los mismos 4 escenarios y las mismas condiciones para los conjuntos de entrenamiento y test.

En la figura III.6.13, se muestra una comparación de los resultados obtenidos para el escenario 3, utilizando la fase 1 del algoritmo *Complex-Log* y el algoritmo completo. Se puede ver que la mejora de los resultados es notable y que tal y como se predecía, la fase 2 es necesaria para realizar la clasificación de imágenes rotadas.

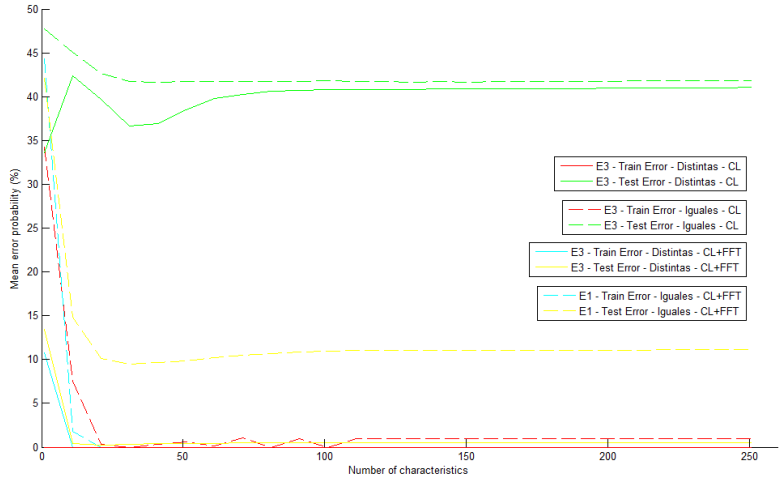


Figura III.6.13: Resultados obtenidos en el escenario 3 utilizando la fase 1 del algoritmo *Complex-Log* y el algoritmo completo.

Después de comprobar la necesidad de aplicar las dos fases del algoritmo *Complex-Log*, se han realizado las mismas simulaciones que en la sección III.6.3.1. Con los resultados obtenidos se ha comprobado que la FFT es necesaria para que la máquina reconozca las imágenes rotadas. Con esta nueva aportación se obtienen los resultados esperados que se comentaron en la sección III.6.3.1.

Como se ha comentado durante el capítulo, los escenarios 1 y 2 eran de los que se esperaban peores resultados, ya que al no aplicar el algoritmo *Complex-Log*, la SVM no es capaz de reconocer las imágenes. Es cierto, que en el Escenario 2, cuando se aumenta el número de muestras rotadas que participan en la fase de entrenamiento, la SVM es capaz de clasificar mejor, pero aun así los porcentajes que se obtienen son altos.

En los escenarios 3 y 4, ya se añade el algoritmo *Complex-Log*, con este algoritmo se obtiene una representación de las imágenes invariantes a rotaciones y escalados. Estas representaciones son utilizadas tanto en el conjunto de entrenamiento como en el de test, lo que hace que la máquina aunque no posea imágenes rotadas en distintos ángulos sea capaz de clasificarlas correctamente. Esto es lo que se puede ver en las figuras III.6.14, III.6.15 y III.6.16.

En cuanto a los vectores soporte, se puede ver en las figuras III.6.17 y III.6.18 que disminuyen con respecto a los obtenidos sin aplicar la FFT, lo cual es lógico, ya que la cantidad de información que necesita la SVM para discriminar entre las clases que intervienen en el problema es menor.

Con los resultados obtenidos, queda demostrada la efectividad de incorporar el algoritmo *Complex-Log* al proceso de clasificación. Esta incorporación permite prescindir de imágenes en el conjunto de entrenamiento y realizar una clasificación con un porcentaje de aciertos alto. Además también se ha concluido la necesidad de utilizar los dos pasos del algoritmo para que la clasificación sea realizada de manera satisfactoria.

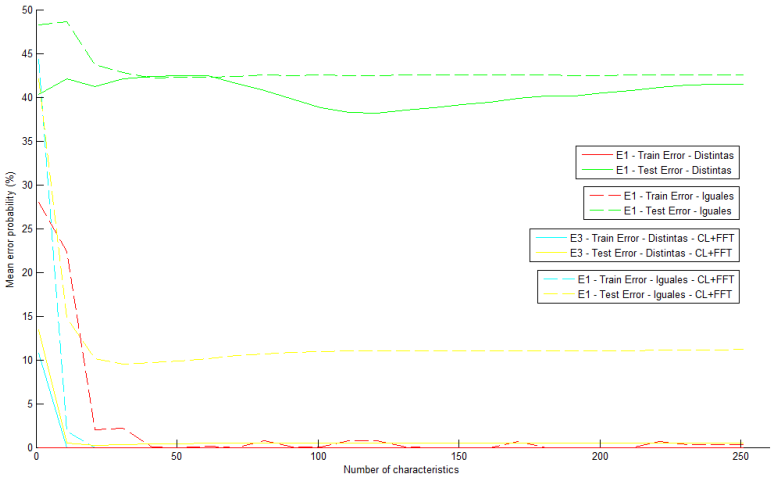


Figura III.6.14: Resultados obtenidos en los escenarios 1 y 3.

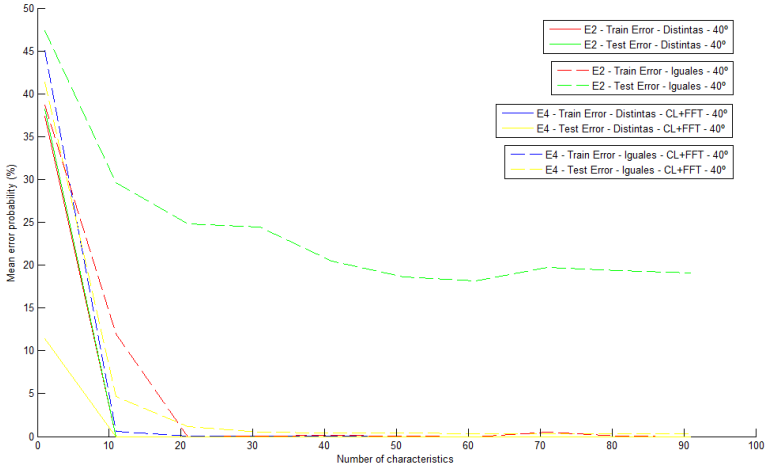


Figura III.6.15: Resultados obtenidos en los escenarios 2 y 4 con saltos de 40° de rotación.

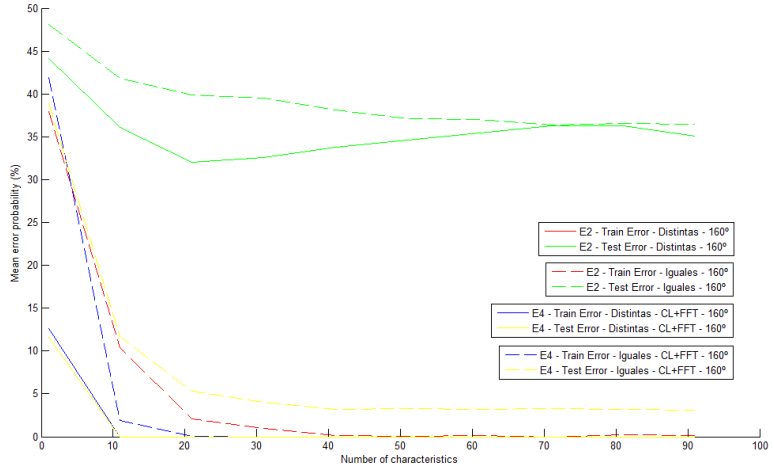


Figura III.6.16: Resultados obtenidos en los escenarios 2 y 4 con saltos de 160° de rotación.

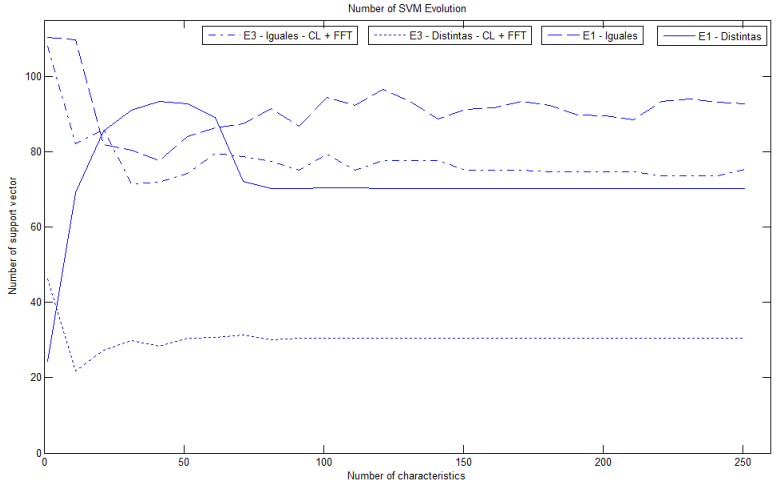


Figura III.6.17: Evolución de los SV en los escenarios 1 y 3.

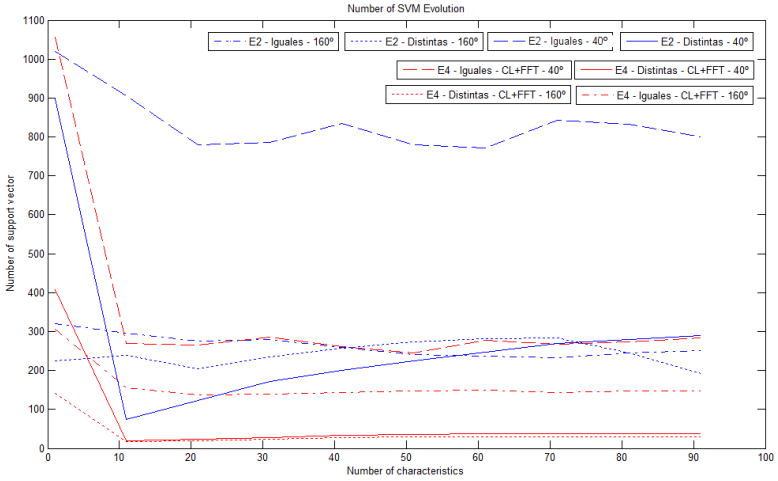


Figura III.6.18: Evolución de los SV en los escenarios 2 y 4 con saltos de 40° y 160° de rotación.

PARTE IV

UN PROYECTO AERONÁUTICO.

Capítulo IV.1

Ciclo de vida de un proyecto de software aeronáutico

Resumen

Este capítulo va a presentar la filosofía de trabajo en un proyecto aeronáutico. La filosofía en “V”es la más utilizada y la que se va a presentar a lo largo del capítulo.

UN proyecto se define como un esfuerzo temporal bajo las premisas de crear un producto o servicio único. Para ello, es necesario aplicar los conocimientos, destrezas, herramientas y técnicas a las actividades del proyecto y además de realizar una correcta gestión del mismo. Todos estos conocimientos se aplican por medio del ciclo de vida del proyecto. El ciclo de vida es un proceso iterativo, en continua realimentación del resto de fases para la detección de fallos en las etapas más tempranas.

Como en todos proyectos, en ingeniería, también se establece un ciclo del vida para los proyectos. Esto permite determinar desde la fase inicial hasta la final, todos los pasos que es necesario seguir. Estos modelos surgen por el hecho de que rectificar en las fases avanzadas es muy costoso y es preferible detectar cuanto antes los posibles fallos existentes en los desarrollos.

En la división de *Defence & Security*, en la sección de ingeniería, el ciclo de vida de los proyectos está especificado en el documento SDF (*System Developement Framework* [Kle08]). Este documento establece la definición y desarrollo del ciclo del vida de un proyecto de ingeniería de software. El proceso de definición se concentra en los procesos técnicos y cómo han de ser gestionados. En él, también se recogen la metodología de trabajo, las *best practices* y las lecciones aprendidas que se han de seguir en futuros desarrollos.

El SDF, tal y como su nombre indica, es un *framework*, lo que significa que no es un plan de proyecto completo, sino que da ciertas consideraciones sobre cómo hay que proceder. De echo da cierta flexibilidad para adaptarlo a las necesidades de cada proyecto. El documento SDF cubre los niveles superiores de los sistemas del proceso de ingeniería, aunque también está integrada la estructura y se tienen en cuenta las interfaces con los niveles inferiores.

En la figura IV.1.1 se define el ciclo de vida de un proyecto aeronáutico a nivel general. En ella se pueden ver las fases de un proyecto de un sistema, subsistema o plataforma. En la figura se puede observar que las tres primeras fases Concepto, Definición y Desarrollo, son las que están cubiertas por el SDF. El proyecto que se está explicando en este documento se puede enmarcar dentro de la fase de “Desarrollo”. En esta fase es en que la que se desarrollan los requisitos del sistema que se fijaron en la fase previa, en este caso al ser un proyecto de *software*, habría que realizar la implementación de los requisitos y realizar las pruebas oportunas para comprobar el funcionamiento de la implementación.

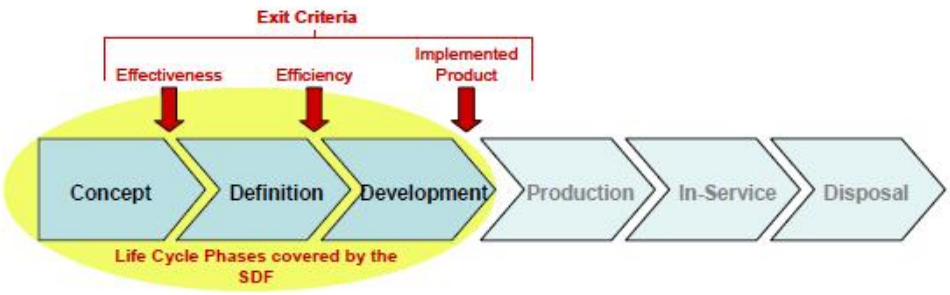


Figura IV.1.1: Ciclo de vida de un proyecto aeronáutico [Kle08].

En los proyectos de ingeniería de software el modelo adoptado para realizar los desarrollos es el llamado, modelo en V. Este modelo está representado en las figuras IV.1.2.

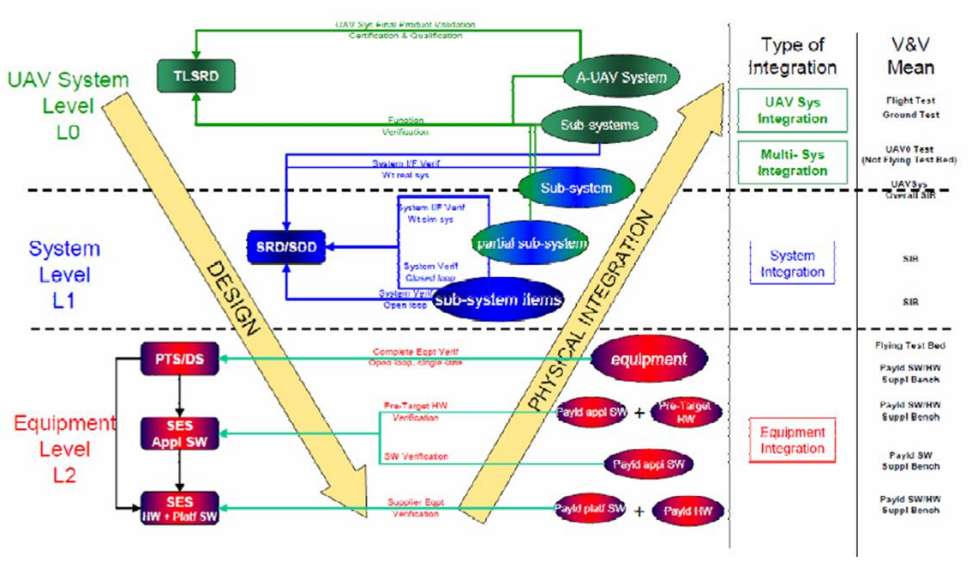


Figura IV.1.2: Modelo en V [MIS09].

En la figura IV.1.2 se muestran las distintas fases por las que ha de pasar un proyecto, además en la figura también están representados los distintos documentos que es necesario pasar al nivel inferior o superior.

El desarrollo de un proyecto puede dividirse en tres partes: la fase de diseño (pata izquierda de la V), fase de desarrollo y pruebas unitarias (pico inferior de la V) y fase de integración (pata derecha de la V). Las tres fases están en continua realimentación, tal y como se puede ver en la figura.

En este proyecto, se parte unos requisitos dados (fase de diseño) y se ocupa del diseño de la aplicación software, la implementación, las pruebas unitarias y la integración en el banco de pruebas, lo que correspondería con el nivel L2 o nivel de equipo en la imagen IV.1.2.

Todo proyecto comienza su desarrollo desde el nivel más alto del sistema, el avión. Una vez aprobado el TLRD (*Top Level Requirements Design*), que contiene los requisitos de alto nivel de la plataforma. En el siguiente paso, se especifica cada uno de los sistemas que compondrán el avión. Estas especificaciones quedan reflejadas en los documentos SDD/SDR (*System Description Document/System Requirements Document*). Estos documentos son pasados a los niveles inferiores, los cuales los analizan y realizan el diseño de las aplicaciones *software* que es necesario implementar o de los diseños *hardware* que hay que realizar. Una vez especificadas las aplicaciones *software* o el *hardware* necesarios, comienza la fase de implementación y desarrollo. Es fundamental que los requisitos estén correctamente redactados y comprendidos por las partes implicadas, especificadores, desarrolladores y testeadores, ya que son contra ellos, contra los que se va a realizar el proceso de verificación.

Una vez realizadas las implementaciones y desarrollos, tanto *software* como *hardware*, los desarrolladores entregan la implementación (*release*) a los ingenieros de test. Aquí el proyecto se encuentra al principio de la parte inferior de la pata derecha de la V.

Para cada nivel de esta nueva fase, se diseña una batería de pruebas específica. En primer lugar se realizan pruebas unitarias para validar que se cumplen todos los requisitos especificados. Hay que puntualizar, que el programador previamente a la entrega de la *release* ha tenido que realizar sus propias pruebas para verificar el correcto funcionamiento de la implementación.

Las baterías de pruebas se realizan contra los requisitos especificados en un inicio, para ello se pueden utilizar herramientas que facilitan el trazado de las pruebas contra ellos, por ejemplo el

programa Doors [MIS09].

En el caso de encontrar algún error en el desarrollo, es necesario comunicarlo a los desarrolladores y volver a revisar todo el desarrollo, corregirlo, y comenzar desde el principio a realizar las pruebas de nuevo. Este proceso se realiza hasta que todas las pruebas sean pasadas satisfactoriamente.

En esta fase de integración, el desarrollo realizado se va integrando nivel a nivel, primero como parte de un subsistema, a continuación como parte de un sistema hasta llegar a formar parte de una plataforma. Ésta es una de las fases más importantes del proyecto.

La integración se define como la combinación de dos o más sistemas o subsistemas para que trabajen juntos funcional y efectivamente. Se entiende sistema como un grupo de partes relacionadas que trabajan juntas para conseguir un propósito común en el avión. Por su parte, subsistema se entiende como una parte del sistema con su propia funcionalidad y propósito específico.

Para que la integración sea posible, en las fases previas es muy importante realizar un ejercicio de definición de interfaces entre las distintas áreas implicadas en el diseño del avión, y que exista una comunicación fluida y continuada durante todo el desarrollo del proyecto.

La integración se puede realizar a distintos niveles tal y como se enumeran a continuación:

1. **Nivel de misión.** Es necesario interactuar con otros aviones y el entorno en el que se desarrolla la misión.
2. **Nivel de plataforma.** Todos los sistemas del avión han de colaborar para que el avión realice su misión más importante, volar.
3. **Nivel de sistema.** Todos los sistemas interactúan con el resto de sistemas y subsistemas del avión para desarrollar su función específica.
4. **Nivel de componente.** Todos los componentes juntos forman un subsistema del avión, tanto a nivel hardware como a nivel software.

En el caso de este proyecto, como se ha dicho anteriormente, se ubica dentro del nivel L2 de la figura IV.1.2. En el proyecto como ya se ha comentado, se han planteado dos fases de pruebas: una de pruebas unitarias y otra de test en un RIG, ver II.5. Una vez realizadas estas pruebas se quiere plantear si es viable realizar una integración software en el sistema que controla el sensor EO y una integración de los sistemas a nivel de plataforma. Éstas se ha desarrollado en los capítulos posteriores, IV.2 y IV.3

La figura IV.1.3 muestra los procesos que se realizan en paralelo al proceso en V de un proyecto. Desde el inicio del mismo es fundamental tener presente los procesos mostrados en la figura, los más importantes son los proceso de certificación y verificación.

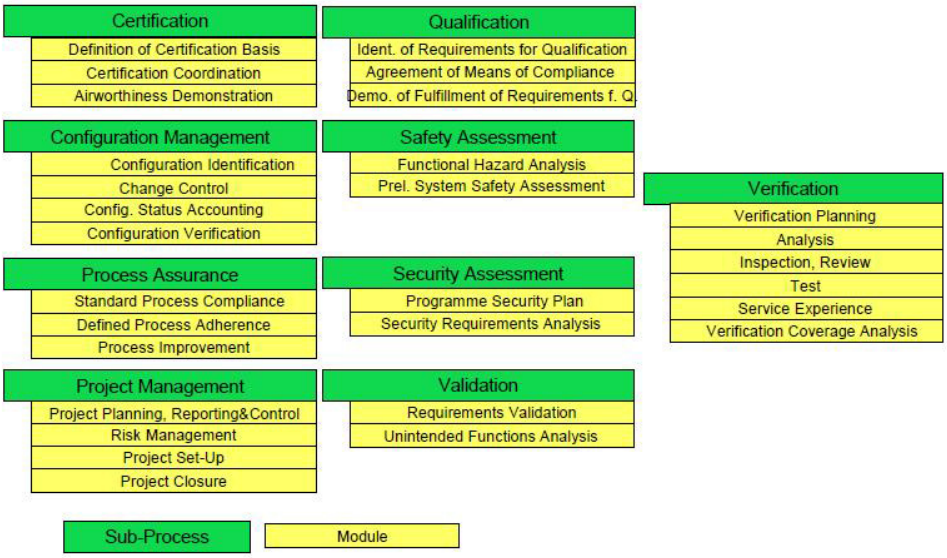


Figura IV.1.3: Módulos auxiliares [Kle08].

En el proceso de certificación de *software* embarcado hay una norma que es fundamental conocer y cumplir, **DO-178B "*Software Considerations in Airbone Systems and Equipment Certification*" [Chi02] y [Mes07].**

La FAA y EASA (*European Aviation Safety Agency*), organismos reguladores americano y europeo, obligan el cumplimiento de esta norma como vía para demostrar que el *software* desarrollado cumple con los requisitos de seguridad establecidos en la norma FAR/CS 25.1309 (estándar estadounidense/europeo). Esta norma es parte de los estándares de aeronavegabilidad de aviones de transportes, en concreto, esta subparte es para los equipos, sistemas e instalaciones.

En la norma DO-178B se establecen los distintos niveles de seguridad que se le ha de asignar al *software* en función de la criticidad de la función que desarrollan. Define los planes que es necesario establecer para el desarrollo de *software*, los estándares que hay que cumplir, los planes de verificación, el entorno de desarrollo... El *software* embarcable está muy controlado debido a la criticidad que puede suponer que se produzca un fallo inesperado durante su ejecución. Por este motivo, no es posible utilizar cualquier lenguaje de programación, se necesario que el lenguaje esté certificado. Además no todas las funcionalidades que ofrecen los lenguajes está disponibles en las versiones de desarrollo de *software* embarcado, están muy limitadas. Para implementar *software* embarcado, los lenguajes que se suelen utilizan son ADA y C. En el caso de C, se utiliza el estándar MISRA-C que especifica las reglas de programación que hay que utilizar para el desarrollo de código embarcable.

Capítulo IV.2

Integración a nivel de componente software

Resumen

Los recursos de memoria y capacidad computacional disponibles en un computador de un avión son limitados. Por este motivo es fundamental conocer la eficiencia de un algoritmo, tanto en memoria como en coste computacional. Con este análisis se podrá determinar la viabilidad o no de embarcar los algoritmos desarrollados en el UAV o por contra la necesidad de hacerlo en la estación de tierra.

En cualquier desarrollo software es fundamental conocer la eficiencia del mismo, es decir el aprovechamiento de los recursos computacionales. Para conocer la eficiencia de un algoritmo se han de estudiar dos factores:

- **La complejidad espacial** o cantidad de memoria que consume.
- **La complejidad temporal** o el tiempo que es necesario para resolver el problema.

Estos dos factores indican la complejidad computacional de un algoritmo. En ocasiones, estas dos métricas pueden coincidir, pero por lo general no es así, de hecho se pondrán en contraposición y será necesario llegar a un compromiso entre ambas.

En el caso de software aeronáutico es imprescindible conocer estos dos factores, debido a que los recursos disponibles en los computadores del avión son limitados. Para poder realizar estas medidas de manera independiente a la plataforma que se desea utilizar, se ha desarrollado una técnica de análisis de algoritmos. Esta técnica indica la evolución del gasto de tiempo y memoria en función del tamaño de los valores de entrada.

En el caso que nos ocupa con este análisis se podrá determinar la viabilidad de embarcar el software desarrollado en el avión, o por el contrario, la necesidad de ejecutarlo en la estación base.

El análisis de algoritmos [[MG02], [HS] y [Duc07]] estudia, desde el punto de vista teórico, los recursos computacionales que necesita la ejecución de un programa de ordenador: su eficiencia. Por ejemplo, el ancho de banda, el tiempo de computación y el espacio en memoria.

Para realizar estas medidas, se parte de la suposición de que está habilitado el acceso aleatorio a memoria, y que cuesta una cantidad fija de tiempo tanto el acceso a memoria como al guardar un elemento en ella.

Dado un algoritmo, la primera tarea es determinar qué tipo de operaciones se utilizan y cuál es su coste relativo. Estas operaciones incluyen las operaciones básicas aritméticas en enteros, sumas, restas, multiplicaciones y divisiones. También pueden encontrarse, operaciones aritméticas con enteros de punto flotante, comparaciones, asignaciones y llamadas a procedimientos. Todas las operaciones descritas tardan una cantidad fija de tiempo en ejecutarse, están acotadas por una constante.

La segunda fase, consiste en determinar un conjunto de datos numéricos suficiente que englobe en él todos los posibles patrones de comportamiento del algoritmo bajo análisis. Esta tarea requiere la comprensión del trabajo que realiza el algoritmo y es necesario conocer las configuraciones del algoritmo que producen los siguientes comportamientos,

- Caso peor. Calcula el tiempo máximo necesario para un problema.
- Caso medio. Calcula el tiempo esperado para un problema cualquiera. Requiere establecer una distribución estadística.
- Caso mejor. Calcula el tiempo mínimo necesario para un problema. Este método es el más engañoso.

Por lo general se suele realizar el estudio de los algoritmos en función del coste en el peor caso. Esto se debe a que proporciona garantías sobre el coste máximo del algoritmo, este valor nunca será excedido y además es el más sencillo de calcular.

Para realizar un análisis completo de un algoritmo, es necesario distinguir entre dos fases: el análisis a priori y el test a posteriori. En el análisis a priori se obtiene una función (en función

de los parámetros más relevantes) que limita el tiempo del algoritmo de computación. En los test a posteriori se recogen las estadísticas sobre el consumo de tiempo y espacio de los algoritmos mientras se ejecutan.

La métrica que se desea determinar es el tiempo total que cada sentencia tarda en ejecutarse, dado un estado inicial de los datos de entrada. Este cálculo requiere dos elementos de información: la frecuencia de la sentencia (número de veces que es ejecutada) y el tiempo que lleva cada ejecución. El producto de estos dos números es el tiempo total.

Como se ha comentado en la introducción, es importante aislar el análisis del algoritmo del computador utilizado y del lenguaje de programación. El análisis a priori elimina estas limitaciones determinando la frecuencia de cada sentencia, lo que puede determinarse directamente desde el algoritmo.

IV.2.1 Notación asintótica

El análisis a priori para calcular el tiempo de computación ignora todos los factores de dependencia en cuanto al tipo de computador utilizado o el lenguaje de programación y se concentra en determinar el orden de magnitud de la frecuencia de ejecución de las sentencias del algoritmo.

Existen distintos tipos de notación, O, Ω y Θ. Cada una de ellas se utiliza para calcular una métrica distinta

- **Notación** O**.** Se utiliza para manejar la cota superior del tiempo de ejecución.
- **Notación** Ω**.** Se utiliza para manejar la cota inferior del tiempo de ejecución.
- **Notación** Θ**.** Se utiliza para indicar el orden exacto de complejidad.

IV.2.1.1 Notación O

Dada una función $f : N \rightarrow R^+$ la clase $O(g(n))$ (se lee, O de g de n), se define por:

$f(n) = O(g(n)) \doteq \{t : N \rightarrow R^+ | \exists c \epsilon R^+, \exists n_0 \epsilon N : \forall n \geq n_0 \quad f(n) \leq cg(n)\}$

Se utiliza esta métrica cuando se quiere determinar el tiempo de computación, $f(n)$, de un algoritmo. La variable n puede representar el número de entradas o salidas, la suma o la magnitud de alguna de ellas. La función $f(n)$ depende del computador que se utilice, por este motivo el análisis a priori calcula una función $g(n)$ tal que $f(n) = O(g(n))$.

Cuando se dice que el tiempo de computación de un algoritmo es $O(g(n))$, significa que si ejecutamos el algoritmo en cualquier computador bajo el mismo tipo de datos pero incrementando los valores de n, el resultado será siempre menos que una constante c veces $g(n)$.

Cuando se determina el orden de magnitud de $f(n)$ siempre se tratará de obtener el menor $g(n)$ tal que $f(n) = O(g(n))$. Esto determina una cota superior de crecimiento de la función $g(n)$.

IV.2.1.1.1 Órdenes de magnitud

El orden de magnitud de una sentencia de un algoritmo hace referencia a su frecuencia de ejecución, mientras el orden de magnitud de un algoritmo se refiere a la suma de las frecuencias de todas las sentencias que lo forman. El orden de magnitud de un algoritmo es representado por $O(f(n))$, siendo la función $f(n)$ más sencilla. El análisis a priori de un algoritmo consiste en determinar el orden de magnitud del algoritmo.

$O(1)$	orden constante
$O(\log n)$	orden logarítmico
$O(n)$	orden lineal
$O(n \log n)$	
$O(n^2)$	orden cuadrático
$O(n^3)$	orden cúbico
$O(n^a)$	orden polinomial ($a > 2$)
$O(a^n)$	orden exponencial ($a > 2$)
$O(n!)$	orden factorial

Tabla IV.2.1: Ordenes de magnitud de la notación O.

A continuación se muestra la relación entre las distintas cotas.

$O(1) < O(\log n) < O(n) < O(n \log(n)) < O(n^2) < O(n^3) < O(2^n)$

$\Omega(1) > \Omega(\log n) > \Omega(n) > \Omega(n \log(n)) > \Omega(n^2) > \Omega(n^3) > \Omega(2^n)$

$O(1)$ significa que el número de ejecuciones de las operaciones base es fijo y por tanto el tiempo total está acotado por una constante. Las primeras 6 cotas tienen la propiedad común de estar acotadas por un polinomio. Gráficamente puede ver en la figura IV.2.1.

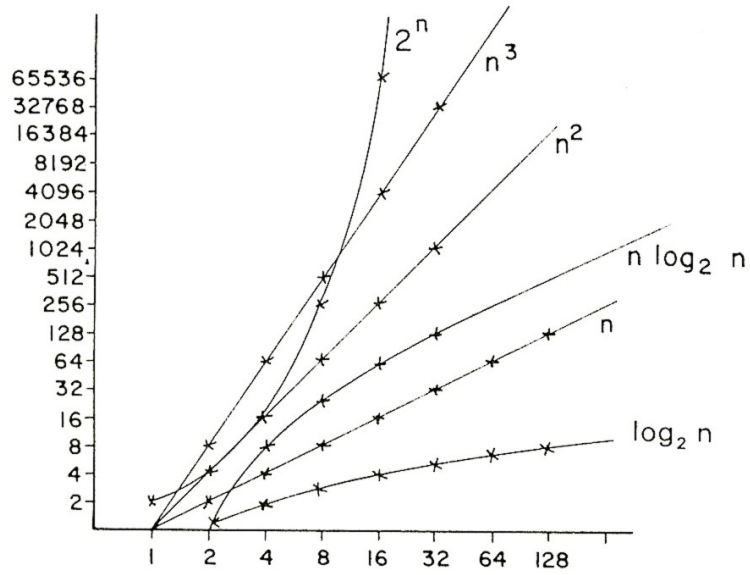


Figura IV.2.1: Tasa de crecimiento de las distintas cotas [HS].

Cualquier algoritmo que esté acotado inferiormente por $O(2^n)$ o $\Omega(2^n)$ se dice que requiere un coste computacional exponencial.

IV.2.1.2 Notación Ω

Dada una función $f : N \to R^+$ la clase $\Omega(g(n))$ (se lee, Omega de g de n) y se define por:

$f(n) = \Omega(g(n)) \doteq \{t : N \to R^+ | \exists c \epsilon R^+, \exists n_0 \epsilon N : \forall n \geq n_0 \quad f(n) \geq cg(n)\}$

IV.2.1.3 Notación Θ

Dada una función $f : N \rightarrow R^+$ la clase $\Theta(g(n)) = O(g(n)) \bigcap \Omega(g(n))$ (se lee, Zita de g de n)

IV.2.1.4 Reglas de simplificación

- **Regla de la suma.** $\Theta(f) + \Theta(g) = \Theta(f + g) = \Theta(\text{máx}(f, g))$.
- **Regla del producto.** $\Theta(f)\Theta(g) = \Theta(fg)$
- Si $f(n) \subset O(g(n)), g(n) \subset O(h(n)) \rightarrow f(n) \subset O(h(n))$
- $O(kg(n)) = O(g(n)) \text{si} k > 0$
- $cO(f(n)) = O(f(n))$
- $O(O(f(n))) = O(f(n))$
- $f(n)O(g(n)) = O(f(n)g(n))$
- $O(f(n)) + O(g(n)) = O(f(n)) \text{si} O(g(n)) \subseteq O(f(n))$

IV.2.1.5 Ejemplo 1: Sumatorio de una serie aritmética

Se desea calcular el coste computacional del sumatorio de una serie aritmética. A continuación se lista el código que se desea analizar y la métrica de las líneas implicadas en el cálculo del coste computacional.

```
1  double suma aritmetica(double a1, double d, int n){
2    double s, an;
3    int i;
4    an = a1;
5    s = a1;
6    for (i=2; i<=n; i++) {
7       an += d;
8       s += an;
9    }
10   return s;
11 }
```

línea	Métrica
línea 4	$O(1)$
línea 5	$O(1)$
línea 6	$O(N)$
línea 7	$O(1)$, $O(N)$ veces
línea 8	$O(1)$, $O(N)$ veces
línea 10	$O(1)$

Tabla IV.2.2: Tiempos de ejecución de cada línea.

El coste temporal de este método es $5 \cdot O(1) + O(n) + 2 \cdot O(1) \cdot O(n)$. Simplificando, el coste es $O(n)$.

IV.2.1.6 Ejemplo 2: Producto de matrices

El programa que se lista a continuación realiza un producto de dos matrices de nxm y sxr, siendo $m = s$.

```
1  void producto matrices(double a[n][m], double b[s][r], double c[n][r]){
2    if(m==s){
3      int i, j, k;
4      for (i=0; i<n; i++) {
5        for (j=0; j<r; j++) {
6          c[i][j] = 0.0;
7          for (k=0; k<m; k++)
8            c[i][j] += a[i][k] * b[k][j];
9        }
10     }
11   }
12 }
```

línea	Métrica
línea 4	$O(n)$
línea 5	$O(r)$, $O(n)$ veces
línea 6	$O(1)$, $O(n * r)$ veces
línea 7	$O(m)$, $O(n * r)$ veces
línea 8	$O(1)$, $O(n * r * m)$ veces

Tabla IV.2.3: Tiempos de ejecución de cada línea.

El coste temporal de este método es $O(n * r * m)$.

IV.2.2 Análisis temporal de los algoritmos

Para recapitular, se van a recordar brevemente en que consisten los dos sub-proyectos desarrollados.

El proyecto desarrollado en EADS consiste en la implementación de dos algoritmos para el manejo y control del sensor EO que lleva en la parte inferior del fuselaje el UAV. El algoritmo de geolocalización, permite conocer la posición del lugar donde está apuntando la cámara y, el algoritmo de geoapuntamiento, permite calcular la actitud de la cámara para que ésta apunte a un objetivo determinado.

El proyecto desarrollado en la Universidad Carlos III, consiste en el desarrollo de un clasificador de imágenes utilizando las SVM. En este subproyecto se puede dividir en dos partes, una parte offline que consiste en entrenar la SVM con una librería de imágenes y una parte online que consiste en clasificar la muestra capturada.

Como se ha dicho el entrenamiento es un proceso offline, en el se realiza un preprocesado de las imágenes, una extracción de características, con la que se calcula la matriz de características y para finalizar, se entrena la SVM. Todo este proceso se hace en tierra y con anterioridad a que el avión vuele. De esta fase se obtienen como salida la matriz de características y los vectores soporte de la máquina SVM entrenada. Estas salidas se deberán cargar en el UAV previamente a su salida a volar. Únicamente se realizará la evaluación del proceso online.

IV.2.2.1 Control de sensores EO

Los algoritmos implementados se explican en detalle en el capítulo II.2. A continuación se muestran los pasos básicos.

IV.2.2.1.1 Geo-localización

El algoritmo de geo-localización calcula la posición del TGT al que está apuntando la cámara (para más información ver la sección II.2.2).

1. Calculo de AC_{ECEF}

$$\overrightarrow{AC_{ECEF}} = \begin{pmatrix} \cos(\lambda_{AC})\cos(\varphi)\left(\left(\frac{a^2}{\sqrt{a^2\cos^2(\varphi)+b^2\sin^2(\varphi)}}\right)+h_{AC}\right) \\ \sin(\lambda_{AC})\cos(\varphi)\left(\left(\frac{a^2}{\sqrt{a^2\cos^2(\varphi)+b^2\sin^2(\varphi)}}\right)+h_{AC}\right) \\ \sin(\varphi_{AC})\left(\frac{b^2}{a^2}+h_{AC}\right) \end{pmatrix} \quad \text{(IV.2.1)}$$

2. Cálculo de LOS_{ECEF}

$$\overrightarrow{LOS_{LOCAL}} = C^{NED}_{LOCAL} C^{BODY}_{NED} \overrightarrow{LOS_{LOCAL\_BODY}}$$

$$= C^{NED}_{LOCAL} C^{BODY}_{NED} (\overrightarrow{LOS_{BODY}} - \overrightarrow{T_{EO\_CG\_BODY}})$$

$$= C^{NED}_{LOCAL} C^{BODY}_{NED} \left(\begin{pmatrix} \sin(\alpha)|LOS|\cos(\beta) \\ \cos(\alpha)|LOS|\cos(\beta) \\ \sin(\beta)|LOS| \end{pmatrix} - \begin{pmatrix} X_{EO\_CG} \\ Y_{EO\_CG} \\ Z_{EO\_CG} \end{pmatrix} \right) \quad \text{(IV.2.2)}$$

3. Cálculo de TGT_{ECEF}

$$\overrightarrow{TGT_{ECEF}} = \overrightarrow{LOS_{LOCAL}} + \overrightarrow{AC_{ECEF}} =$$

$$C^{NED}_{LOCAL} C^{BODY}_{NED} \begin{pmatrix} \sin(\alpha)|LOS|\cos(\beta) - X_{EO\_CG} \\ \cos(\alpha)|LOS|\cos(\beta) - Y_{EO\_CG} \\ \sin(\beta)|LOS| - Z_{EO\_CG} \end{pmatrix} +$$

$$+ \begin{pmatrix} \cos(\lambda_{AC})\cos(\varphi)\left(\left(\frac{a^2}{\sqrt{a^2\cos^2(\varphi)+b^2\sin^2(\varphi)}}\right) + h_{AC}\right) \\ \sin(\lambda_{AC})\cos(\varphi)\left(\left(\frac{a^2}{\sqrt{a^2\cos^2(\varphi)+b^2\sin^2(\varphi)}}\right) + h_{AC}\right) \\ \sin(\varphi_{AC})\left(\frac{b^2}{a^2} + h_{AC}\right) \end{pmatrix} \quad \text{(IV.2.3)}$$

4. Cálculo de la latitud, longitud y altura del TGT

$$\overrightarrow{TGT_{LLA}} = \begin{pmatrix} \lambda_{TGT} \\ \varphi_{TGT} \\ h_{TGT} \end{pmatrix} = \begin{pmatrix} atan2(Y\_TGT_{ECEF}, X\_TGT_{ECEF}) \\ atan2\left(Z_{ECEF} + \frac{e^2 a^2 \sin^3(\zeta)}{b}, \varepsilon - e^2 a\cos^3(\zeta)\right) \\ \frac{\varepsilon}{\cos\varphi} - r_T \end{pmatrix} \quad \text{(IV.2.4)}$$

donde, $\varepsilon = \sqrt{X^2_{ECEF} + Y^2_{ECEF}}$; $\zeta = atan2(a * Z_{ECEF}, b\varepsilon)$; $r_T = \frac{a}{\sqrt{1-e^2\sin^2(\varphi)}}$

Tras unificar todos los pasos descritos, se desea realizar todas las operaciones posibles para que el cálculo quede lo más unificado posible y así reducir el tiempo y recursos computacionales utilizados. Para ello, se realizan los siguientes pasos.

1. Las matrices C^{NED}_{LOCAL} y C^{BODY}_{NED} se multiplican para crear una única matriz C^{BODY}_{LOCAL} y así se ahorran recursos computacionales.

$$\overrightarrow{TGT_{ECEF}} = C^{BODY}_{LOCAL} \begin{pmatrix} \sin(\alpha)|LOS|\cos(\beta) - X_{EO\_CG} \\ \cos(\alpha)|LOS|\cos(\beta) - Y_{EO\_CG} \\ \sin(\beta)|LOS| - Z_{EO\_CG} \end{pmatrix} +$$

$$+ \begin{pmatrix} \cos(\lambda_{AC})\cos(\varphi)\left(\left(\frac{a^2}{\sqrt{a^2\cos^2(\varphi)+b^2\sin^2(\varphi)}}\right) + h_{AC}\right) \\ \sin(\lambda_{AC})\cos(\varphi)\left(\left(\frac{a^2}{\sqrt{a^2\cos^2(\varphi)+b^2\sin^2(\varphi)}}\right) + h_{AC}\right) \\ \sin(\varphi_{AC})\left(\frac{b^2}{a^2} + h_{AC}\right) \end{pmatrix} \quad \text{(IV.2.5)}$$

2. Debido a la existencia de traslaciones, es muy útil la utilización de coordenadas homogéneas (ver sección II.1.7). Como ya se explicaron, las coordenadas homogéneas permiten el tratamiento de rotaciones y traslaciones como productos de matrices, con sólo añadir una cuarta coordenada al problema. Utilizando las coordenadas homogéneas, el vector $\overrightarrow{TGT_{ECEF}}$ quedaría,

$$\overrightarrow{TGT_{ECEF}} = \left(C^{BODY}_{LOCAL} \begin{pmatrix} \sin(\alpha)|LOS|\cos(\beta) - X_{EO\_CG} \\ \cos(\alpha)|LOS|\cos(\beta) - Y_{EO\_CG} \\ \sin(\beta)|LOS| - Z_{EO\_CG} \end{pmatrix} \right) +$$

$$\begin{pmatrix} \cos(\lambda_{AC})\cos(\varphi)\left(\left(\frac{a^2}{\sqrt{a^2\cos^2(\varphi)+b^2\sin^2(\varphi)}}\right)+h_{AC}\right) \\ \sin(\lambda_{AC})\cos(\varphi)\left(\left(\frac{a^2}{\sqrt{a^2\cos^2(\varphi)+b^2\sin^2(\varphi)}}\right)+h_{AC}\right) \\ \sin(\varphi_{AC})\left(\frac{b^2}{a^2}+h_{AC}\right) \end{pmatrix} =$$

$$\begin{pmatrix} C^{BODY}_{LOCAL} & 0 \\ 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 & \sin(\alpha)|LOS|\cos(\beta)-X_{EO\_CG} \\ 0 & 1 & 0 & \cos(\alpha)|LOS|\cos(\beta)-Y_{EO\_CG} \\ 0 & 0 & 1 & \sin(\beta)|LOS|-Z_{EO\_CG} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$* \begin{pmatrix} 1 & 0 & 0 & \cos(\lambda_{AC})\cos(\varphi)\left(\left(\frac{a^2}{\sqrt{a^2\cos^2(\varphi)+b^2\sin^2(\varphi)}}\right)+h_{AC}\right) \\ 0 & 1 & 0 & \sin(\lambda_{AC})\cos(\varphi)\left(\left(\frac{a^2}{\sqrt{a^2\cos^2(\varphi)+b^2\sin^2(\varphi)}}\right)+h_{AC}\right) \\ 0 & 0 & 1 & \sin(\varphi_{AC})\left(\frac{b^2}{a^2}+h_{AC}\right) \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{(IV.2.6)}$$

Después de estas operaciones bastaría con calcular la latitud, longitud y altura. Éstas son operaciones básicas cuyo coste computacional es $O(1)$. Al final el algoritmo se reduce a una multiplicación de 3 matrices de dimensiones 4*4. Aplicando las reglas de simplificación vistas con anterioridad en la sección IV.2.1.4, el coste computacional del algoritmo es $O(N^3) + O(N^3) = (N^3)$. La conclusión es que el coste computacional teórico es constante, $O(1)$, ya que N siempre es 4.

IV.2.2.1.2 Geo-apuntamiento

El algoritmo de geo-apuntamiento calcula la actitud del sensor EO para apuntar a un determinado TGT (para más información ver la sección II.2.3).

1. Calculo de AC_{ECEF}

$$\overrightarrow{AC_{ECEF}} = \begin{pmatrix} \cos(\lambda_{AC})\cos(\varphi)\left(\left(\frac{a^2}{\sqrt{a^2\cos^2(\varphi)+b^2\sin^2(\varphi)}}\right)+h_{AC}\right) \\ \sin(\lambda_{AC})\cos(\varphi)\left(\left(\frac{a^2}{\sqrt{a^2\cos^2(\varphi)+b^2\sin^2(\varphi)}}\right)+h_{AC}\right) \\ \sin(\varphi_{AC})\left(\frac{b^2}{a^2}+h_{AC}\right) \end{pmatrix} \quad \text{(IV.2.7)}$$

2. Calculo de TGT_{ECEF}

$$\overrightarrow{TGT_{ECEF}} = \begin{pmatrix} \cos(\lambda_{TGT})\cos(\varphi)\left(\left(\frac{a^2}{\sqrt{a^2\cos^2(\varphi)+b^2\sin^2(\varphi)}}\right)+h_{TGT}\right) \\ \sin(\lambda_{TGT})\cos(\varphi)\left(\left(\frac{a^2}{\sqrt{a^2\cos^2(\varphi)+b^2\sin^2(\varphi)}}\right)+h_{TGT}\right) \\ \sin(\varphi_{TGT})\left(\frac{b^2}{a^2}+h_{TGT}\right) \end{pmatrix} \quad \text{(IV.2.8)}$$

3. Cálculo de LOS_{LOCAL}

$$\overrightarrow{LOS_{LOCAL\_BODY}} = (C^{NED}_{BODY}C^{LOCAL}_{NED}(\overrightarrow{TGT_{ECEF}} - \overrightarrow{AC_{ECEF}})) + (\overrightarrow{DIST_{EO\_CG}})$$

$$= \left(C_{BODY}^{NED} C_{NED}^{LOCAL} \begin{pmatrix} X_{TGT_{ECEF}} \\ Y_{TGT_{ECEF}} \\ Z_{TGT_{ECEF}} \end{pmatrix} - \begin{pmatrix} X_{AC_{ECEF}} \\ Y_{AC_{ECEF}} \\ Z_{AC_{ECEF}} \end{pmatrix} \right) + \begin{pmatrix} X_{EO\_CG} \\ Y_{EO\_CG} \\ Z_{EO\_CG} \end{pmatrix} \quad \text{(IV.2.9)}$$

4. Cálculo de los ángulos de Azimut y Elevación del sensor EO

$$\boxed{Azimut = \alpha = atan2(Y_{LOS_{BODY}}, X_{LOS_{BODY}})} \quad \text{(IV.2.10)}$$

$$\boxed{Elevación = \beta = atan2(Z_{LOS_{BODY}}, \sqrt{X_{LOS_{BODY}}^2 + Y_{LOS_{BODY}}^2})} \quad \text{(IV.2.11)}$$

Tras unificar todos los pasos descritos, se desea realizar todas las operaciones posibles para que el cálculo quede lo más unificado posible y así reducir el tiempo y recursos computacionales utilizados. Para ello, se realizan los siguientes pasos.

1. Las matrices C_{BODY}^{NED} y C_{NED}^{LOCAL} se multiplican para crear una única matriz C_{BODY}^{LOCAL} y así ahorrar recursos computacionales.

2. Al igual que en el caso anterior, la utilización de coordenadas homogéneas (ver sección II.1.7) permite tratar las rotaciones y traslaciones, como productos de matrices, con sólo añadir una cuarta coordenada al problema. Utilizando las coordenadas homogéneas, el vector $\overrightarrow{LOS_{LOCAL}}$ sería,
$\overrightarrow{LOS_{LOCAL\_BODY}} = (C_{BODY}^{LOCAL}(\overrightarrow{TGT_{ECEF}} - \overrightarrow{AC_{ECEF}})) + (\overrightarrow{DIST_{EO\_CG}}) =$

$$\left(\begin{pmatrix} C_{BODY}^{LOCAL} & 0 \\ 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 & (X_{TGT_{ECEF}} - X_{AC_{ECEF}}) \\ 0 & 1 & 0 & (Y_{TGT_{ECEF}} - Y_{AC_{ECEF}}) \\ 0 & 0 & 1 & (Z_{TGT_{ECEF}} - Z_{AC_{ECEF}}) \\ 0 & 0 & 0 & 1 \end{pmatrix} \right) * \begin{pmatrix} 1 & 0 & 0 & X_{EO\_CG} \\ 0 & 1 & 0 & Y_{EO\_CG} \\ 0 & 0 & 1 & Z_{EO\_CG} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{(IV.2.12)}$$

Después de estas operaciones bastaría con calcular los ángulos de azimut y elevación de la cámara. Éstas son operaciones básicas cuyo coste computacional es $O(1)$. Al final, al igual que en el caso del algoritmo de geo-localización, el algoritmo se reduce a una multiplicación de 3 matrices de dimensiones 4*4. Aplicando las reglas de simplificación vistas con anterioridad en la sección IV.2.1.4, el coste computacional del algoritmo es $O(N^3) + O(N^3) = (N^3)$. La conclusión es que el coste computacional teórico es constante, $= (1)$, ya que N siempre es 4.

IV.2.2.2 Clasificador de imágenes

Como se ha explicado en la introducción del capítulo, en la fase del clasificador únicamente hay que tener en cuenta la fase de *test*, por lo que sólo será necesario disponer de la imagen y de la información cargada al UAV antes de comenzar la misión; es decir, la matriz de características de dimensiones $NChar\ x(píxel * píxel)$ (siendo, $NChar$ el número de características), la matriz de vectores soportes de dimensiones $Nsv\ x\ (píxel * píxel)$ (siendo Nsv, el número de vectores soportes de la SVM) y el vector β_i de dimensiones $Nsv\ x\ 1$.

A continuación se detallan los pasos a seguir suponiendo que la imagen es de $n * m$ píxeles:

1. Normalizar la imagen en media y varianza. Para ello se le resta a cada elemento de la matriz la media de la imagen y se divide entre la varianza de la imagen.

   La media y varianza se calculan de la siguiente manera.

$$Media\_imagen = \frac{\left(\sum_{i=1}^{n}\sum_{j=1}^{m} imagen(i,j)\right)}{n*m} \tag{IV.2.13}$$

$$Varianza\_imagen = \frac{\left(\sum_{i=1}^{n}\sum_{j=1}^{m}(imagen(i,j) - media\right)^2}{n*m} \tag{IV.2.14}$$

   El coste computacional del cálculo de la media es $O(n*m)$, mientras el coste computacional del cálculo de la varianza es $O(n*m)$.

   Ahora es necesario restarle a cada píxel de la imagen la media calculada y dividirlo entre la varianza.

$$Imagen\_normalizada = \left(\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{(imagen(i,j) - media)}{varianza}\right) \tag{IV.2.15}$$

   El coste computacional del cálculo de la imagen normalizada es $O(n*m)$.

2. Aplicar el Algoritmo de *Complex-Log* a la imagen capturada.

   - La primera fase consiste en interpolar la imagen capturada a la rejilla creada para este propósito. Como se explicó la interpolación utilizada es la interpolación lineal, la cual consiste en interpolar con los 4 elementos más cercanos a cada uno de los píxeles que forman la imagen. Suponiendo que la imagen capturada tiene una resolución de $x*y$, el coste computacional de calcular la interpolación de una imagen a otra es de orden $O(n*m*4)$. Al finalizar la interpolación las dimensiones de la imagen se mantienen en n x m píxeles.
   - La segunda fase del algoritmo consiste en realizar la FFT de la representación obtenida. Segun la bibliografía consultada [FFT10] el tiempo computacional de realizar la FFT es de $O(n*m)*\log_2(n*m)$, siendo, $N = n*m$.

3. Modificar la representación matricial de la imagen por un vector de píxeles. El coste computacional es $O(m*n)$ ya que es necesario recorrer todas las columnas y a continuación todas las filas colocando cada píxel uno a continuación del anterior. El nuevo vector que se obtiene tiene dimensiones $(n*m)$ x 1

4. Multiplicar la matriz de características de dimensiones $NChar$ x $(n*m)$ por la representación de la imagen normalizada de dimensiones $(n*m)$ x 1. La matriz resultante es de $NChar$ x 1 píxeles. La multiplicación de matrices como se ha visto durante la explicación, tienen un coste computacional de $O(NChar*(n*m))$

5. Evaluar la imagen mediante la máquina de vectores soporte. La evaluación se hace para cada uno de los vectores soporte calculados durante el entrenamiento de dimensiones Nsv x $NChar$. Se considera que se obtienen Nsv vectores soporte.

$$f(x) = sig\left(\sum_{i=1}^{Nsv}\beta_i k(x,x_i)\right) = sig\left(\sum_{i=1}^{Nsv} g(x)\right) \tag{IV.2.16}$$

   siendo:

- $k(x, x_i) = exp(-\gamma(|x * x_i|)^2)$
- Nsv es el número de vectores soporte.
- x son las muestras que se desean evaluar. Del paso anterior se obtiene una matriz de dimensiones $NChar$ x 1
- x_i son los vectores soportes obtenidos durante el entrenamiento de la máquina (subconjunto de las muestras de entrenamiento). La matriz de vectores soporte tiene las dimensiones Nsv x $NChar$
- $\beta_i = \alpha_i y_i$, se obtiene durante la fase de entrenamiento y tiene las dimensiones N_{SV} x 1
  - α_i son los multiplicadores de Lagrange obtenidos durante la fase de entrenamiento.
  - y_i son las etiquetas de las muestras de entrenamiento.
- $\beta_i = \alpha_i y_i$, se obtiene durante la fase de entrenamiento y tiene las dimensiones Nsv x 1
- γ es un valor del núcleo de dimensiones 1x1

Como resultado de este paso se obtendría el signo de la suma que se obtiene de evaluar para vector soporte (Nsv) la siguiente expresión:

$$g(x) = \beta_i(exp(-\gamma(|x * x_i|)^2))) \tag{IV.2.17}$$

El coste computacional de evaluar esta expresión es.

- Evaluación del núcleo $k(x, x_i) = exp(-\gamma(|x * x_i|)^2) \Rightarrow O(NChar)$
- Evaluar la función $g(x) = \beta_i k(x, x_i) \Rightarrow O(1)$

El coste computacional de evaluar $g(x)$ es, $O(NChar)$.

Ahora hay que analizar la ecuación IV.2.16 que consiste en evaluar la función $g(x)$ para los Nsv vectores soporte obtenidos. El coste computacional total de evaluar la ecuación IV.2.16 es $O(NChar * Nsv)$.

Si se suman los órdenes obtenidos en cada uno de los pasos, se obtiene:

$O(n * m)$+$O(n * m)$+$O(n * m * 4)$+$O(m * n) * \log_2(m * n)$+$O(m * m)$+$O(NChar * (n * m))$+$O(NChar * Nsv)$.

Aplicando las reglas de simplificación (ver sección IV.2.1.4) se obtiene que el coste computacional del algoritmo es $O(m * n) * \log_2(m * n)$+$O(NChar * (n * m))$+$O(NChar * Nsv)$, siendo el coste computacional, el coste de mayor orden de los que forman la suma anterior.

IV.2.3 Análisis espacial de los algoritmos

A parte de la memoria para guardar los ejecutables de los algoritmos, es necesario guardar en memoria, la matriz de características de dimensiones $NChar\ x\ (pixel * pixel)$, siendo NChar el número de características; la matriz de vectores soportes de dimensiones $Nsv\ x\ (pixel * pixel)$, siendo Nsv el número de vectores soportes de la SVM y la matriz β_i de dimensiones $Nsv\ x\ (n * m)$.

En cuanto al coste espacial en memoria durante la ejecución, las variables son variables escalares con lo que el consumo de memoria es constante.

IV.2.4 Conclusiones

A continuación, en la tabla IV.2.4 se muestra una tabla con los resultados obtenidos.

Implementación	Recursos temporales
Geo-localización	$O(1)$
Geo-apuntamiento	$O(1)$
Clasificador de imágenes	$O(m * n) * \log_2(m * n)$+$O(NChar * (n * m))$+$O(NChar * Nsv)$

Tabla IV.2.4: Resultados del análisis de algoritmos.

En la tabla IV.2.4 se puede ver que los algoritmos de geo-localización y geo-apuntamiento tienen un coste computacional teórico constante. Debido a este hecho, es necesario realizar una prueba para medir el coste computacional real que tarda en ejecutarse.

Utilizando un ordenador *Hewlett-Packard*, *Compaq Presario C700 Notebook PC*, con un procesador Intel Celeron 560@2.13GHz, con una memoria RAM de 1GB, un sistema operativo de 32 bits y *Windows Vista Home Basic*; se obtiene que el tiempo en ejecutar cada algoritmo es 350 unidades de tiempo.

En el caso del algoritmo de clasificación de imágenes, el coste computacional teórico depende de las variables de entrada, el número de píxeles, el número de características y el número de vectores soporte. Suponiendo el peor caso, que la imagen sea cuadrada, el número de características sea el máximo posible, $píxel * píxel$; y que el número de vectores soporte sea igual al número de características; el orden del algoritmo sería $O(píxel^4)$. Como se puede ver, el tamaño de la imagen es el que al final determina el coste computacional del algoritmo.

En el caso de que el número de píxeles fuera muy bajo, este algoritmo se podría embarcar. En las pruebas realizadas III.6, el valor máximo de vectores soporte era aproximadamente 200 y el número de características 50, medido para imágenes de 16x16 píxeles. Para estos valores máximos es inviable embarcar este algoritmo. Los recursos computacionales que necesita el algoritmo no pueden ser proporcionados por un computador embarcado en un UAV. En la realidad las imágenes que se van a tratar son de alta calidad (1028x1028), lo que aumenta considerablemente el coste computacional necesario para su clasificación.

Estudios realizados en la Universidad Carlos III de Madrid, han demostrado que utilizando un procesador digital de señales [1](DSP, *Digital Signal Processor*), es posible realizar clasificación de imágenes en tiempo real. Para que el algoritmo sea embarcable, el DSP ha de ser certificable, que el DSP esté dedicado a esta tarea, que posea la capacidad computacional para poder realizar todas las operaciones necesarias (los equipos embarcables suelen tener características limitadas) y disponer de un canal de alta capacidad para la transmisión de las imágenes.

Además no se ha tenido en cuenta en el cálculo computacional, que previamente a la ejecución del algoritmo desarrollado es necesario aislar la muestra que se desea clasificar del conjunto de la imagen (fuera del ámbito del proyecto), con lo que los requerimientos computacionales aumentarían.

Por todo esto se ha decidido que el algoritmo de clasificación de imágenes se integre en la estación base y no se embarque en el UAV.

[1]Un DSP es un sistema basado en un procesador o microprocesador que posee un juego de instrucciones, un *hardware* y un software optimizados para aplicaciones que requieran operaciones numéricas a muy alta velocidad. Debido a esto es especialmente útil para el procesado y representación de señales analógicas en tiempo real.

Capítulo IV.3

Integración a nivel de sistemas

Resumen

En este capítulo se ilustra un ejemplo de Integración a nivel de sistemas.

UNa vez realizada la integración a nivel de componentes *software* en el computador que controla el sensor EO; es necesario realizar una integración a nivel de sistemas del UAV. Los algoritmos que se han implementado necesitan información de otros sistemas del UAV para ejecutarse, y los resultados que obtienen han de ser comunicados al resto de sistemas.

Para realizar la integración de los algoritmos se ha definido la existencia de un computador que gestione todas las operaciones relacionadas con la cámara, el computador se ha llamado *Sensor EO Management Computer*. Como se ha venido haciendo durante todo el proyecto, se ha tomado como UAV de referencia el UAV ATLANTE.

En la figura IV.3.1 se pude ver la arquitectura propuesta como ejemplo de integración de los dos subproyectos desarrollados en este Proyecto.

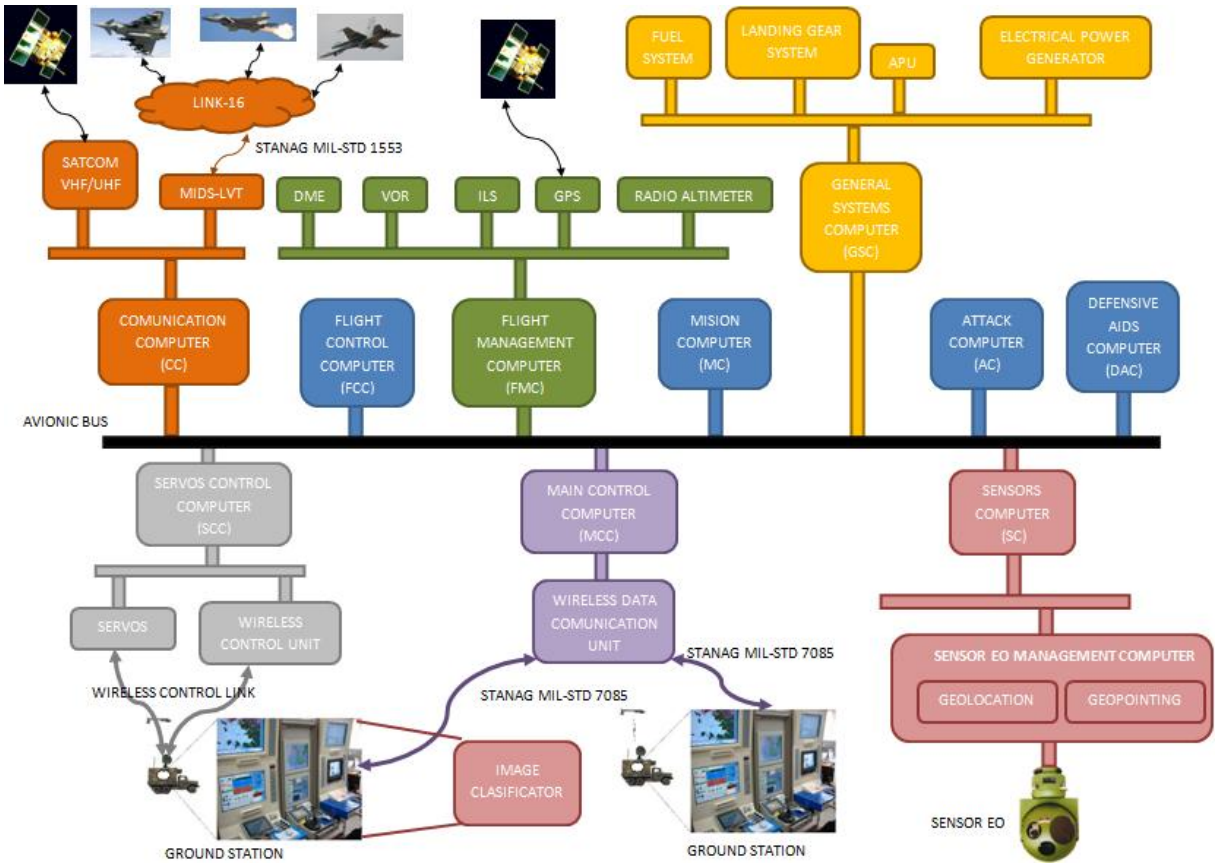


Figura IV.3.1: Ejemplo planteado de integración.

Como se puede ver en la figura IV.3.1, los algoritmos de geo-localización y geo-apuntamiento han sido integrados en un computador llamado *Sensor Computer*, en concreto del computador *sensor EO management computer*, mientra que el algoritmo de clasificación de imágenes se ha integrado en la estación base del UAV, todo esto según lo que se ha concluido en la integración software, ver IV.2.4.

El computador *sensor EO management computer* enviará y recibirá los datos necesarios para la ejecución de sus algoritmos a través del computador *Sensor Computer*. Como entradas necesita:

- Algoritmo de Geolocation
  - Posición del avión, procede del *Flight Control Computer* (FCC).
  - Altura del avión, procede del *Flight Management Computer* (FMC).

  - Actitud del avión, procede del *Flight Management Computer* (FMC).
  - Actitud de la cámara del sensor EO, procede del *Sensor Computer* (SC).
  - Posición del sensor EO respecto del centro de gravedad del avión, procede del *Sensor EO Management Computer*.
  - Distancia al objetivo (designador láser), procede del *Sensor EO Management Computer*.

- Algoritmo de Geopointing

  - Posición del avión, procede del *Flight Control Computer* (FCC).
  - Altura del avión, procede del *Flight Management Computer* (FMC).
  - Actitud del avión, procede del *Flight Management Computer* (FMC).
  - Posición del TGT, procede del *Main Control Computer* (MCC), viene establecido desde la estación de tierra.
  - Posición del sensor EO respecto del centro de gravedad del avión, procede del *Sensor EO Management Computer*.

Las salidas de los algoritmos se puede transmitir a:

- *Main Control Computer (MCC)*: este computador se encarga de transmitir información entre el UAV y la estación de tierra.
- *Comunication Computer (CC)*: Este computador puede transmitir los datos a:
  - Módulo SATCOM (*Satellite Communications*): transmite vía Satélite a otras estaciones de tierra o a aviones amigos, la información requerida.
  - MIDS-LVT (*Multifunctional Information Distribution System-Low Volume Terminal*): transmite la información a otros aviones a través de una red Link-16.
- *Mision Computer*(MC): Se registra la información para analizarla al finalizar la misión e incluir los posibles nuevos resultados en las bases de datos de misión.

A continuación se van a explicar cada uno de los computadores y las comunicaciones que se han representado en la imagen IV.3.1.

Para este proyecto, el computador principal es el ***Sensor EO Management Computer***. Es el encargado de gestionar todas las operaciones del sensor EO: establecer y recibir la actitud de la cámara, recibir la distancia calculada por el designador láser y los parámetros necesarios del resto de sistemas. Además gestiona la recepción de imágenes capturadas por la cámara y su envío a través del SC y éste al MCC a la estación de tierra para su posterior procesado.

Es para este computador, para el que se están desarrollando los algoritmos de este proyecto y en el cual se desean integrar. Como se ha dicho anteriormente, los algoritmos necesitan información de otros sistemas del UAV, será a través de su comunicación con el bus de aviónica como la obtendrá. En concreto, necesita información de los computadores de navegación y de control de vuelo para conocer la actitud y posición del avión. El computador ha de transmitir los resultados obtenidos de la ejecución de los algoritmos al resto de sistemas y a la estación base, para ello también utilizará el bus de aviónica. Los computadores que pueden recibir la información son: el computador de control principal y el computador de comunicaciones.

Para comenzar, es necesario conocer el concepto de STANAG (*STANdarization AGreement*). STANAG es la abreviatura de la NATO para los acuerdos de estandarización, que establece los

procesos, procedimientos, términos y condiciones para los procedimientos comunes militares, técnicas o de equipación entre los países miembros de la alianza. Cada estado de la OTAN ratifica un STANAG y lo aplica dentro de sus propias fuerzas armadas. El propósito, es proporcionar procedimientos comunes de funcionamiento, administrativos y de logística. STANAG también constituye la base para la interoperabilidad entre una amplia variedad de comunicación e información (CIS), los sistemas esenciales para la OTAN y las operaciones aliadas.

Las STANAGs se publican en Inglés y francés, los dos idiomas oficiales de la OTAN, por la agencia de normalización de la OTAN en Bruselas.

Además existe una convención llamada ATA-100, que es una forma de organizar las distintas partes, reparaciones o tipos de sistemas que posee cualquier avión. Puede verse la lista en la referencia [ATA10]

A continuación, se explican cada uno de los computadores implicados en la integración propuesta, viéndolos siempre desde el punto de vista del computador Sensor EO Management Computer. Hay que tener en cuenta, que la información disponible sobre estos sistemas y modos de comunicación entre ellos es muy escasa debido a su calificación de "*Restricted* ".

***Main Control Computer* (MCC).** Es el computador que permite la comunicación entre el UAV y la estación de tierra. Este computador es el enlace entre el UAV y su estación de control. Para que esta comunicación se pueda realizar es necesario implementar el STANAG 7085 "*Interoperable Data Links for Imaging Systems*". Este STANAG permite el envío de datos entre las estaciones bases y los aviones UAVs. Este STANAG trabaja en colaboración con otros STANAGs, como por ejemplo el 4586, uno de los más importantes en los UAVS; ya que define las interfaces necesarias, para que las comunicaciones se puedan realizar. Debido a la naturaleza militar de los STANAG la información sobre ellos tiene restringido el acceso.

***Flight Control Computer* (FCC).** El sistema de control de vuelo de un avión está compuesto por las superficies de control, las conexiones y los mecanismos operacionales necesarios para controlar la dirección de vuelo y conseguir las características aceptables de control y manipulación a través de la envolvente de vuelo, proporcionando continua estabilización automática de las aeronaves, por medio del control por ordenador de las superficies de control. De este sistema, el computador *Sensor EO Management Computer* obtendrá la actitud del avión: ángulo de *pitch*, *roll* y *heading*.

***Navigation Computer* (NC)(ATA 34).** Este sistema está compuesto de un conjunto de técnicas y procedimientos que permiten conducir eficientemente una aeronave a su lugar de destino, asegurando la integridad de los tripulantes, pasajeros y de los que están en tierra. La navegación aérea se basa en la observación del cielo, del terreno y de los datos aportados por los instrumentos de vuelo. De este sistema el computador *Sensor EO Management Computer* obtendrá la posición actual del avión: latitud, longitud y altura.

***Comunication Computer*(CC)(ATA 23).** Este sistema se encarga de todas las comunicaciones radio de la aeronave. En el diagrama de integración propuesto, se ha contemplado la posibilidad de comunicaciones a través de una red Link-16 y un terminal táctico llamado MIDS-LVT o de comunicaciones SATCOM. Esto permitirá al UAV compartir información con otros aviones.

Las comunicaciones con otros aviones a través de la red Link-16 han de ser seguras, ya que la información que se intercambia es de acceso restringido; por eso, es el único sistema utiliza para comunicarse el estándar MIL-STD 1553. Si el UAV actuase como relay de comunicaciones, podría compartir los resultados de la clasificación de las imágenes que

captura con otros aviones. Por ejemplo con un avión MEDEVAC, el UAV podía haber reconocido a los heridos y el avión MEDEVAC (*MEDical EVACuation*) con esa información, sabría el punto exacto donde se encuentran.

- **Link-16.** Es una red de intercambio de datos tácticos utilizada por la OTAN, está definida en la STANAG 5516 y en la MIL-STD 6016.
- **MIL-STD 1553 [oDE98].** MIL-STD-1553 es el estándar militar publicado por el Departamento de Defensa de Estados Unidos, que define las características mecánicas, eléctricas y funcionales de un bus de datos en serie. Fue diseñado en principio para su uso en aviación militar; pero ha terminado por utilizarse habitualmente en subsistemas embarcados de manejo de datos en vehículos espaciales, tanto militares como civiles. Proporciona una interfaz física de línea balanceada dual, una interfaz de red diferencial, multiplexación por división en el tiempo, protocolo de comando/respuesta *half-duplex* y hasta 31 terminales remotos. Actualmente es utilizado por todas las ramas del ejército de Estados Unidos y ha sido adoptado por la OTAN como STANAG 3838 AVS. MIL-STD-1553 está siendo reemplazado en nuevos diseños norteamericanos por FireWire. Una versión de MIL-STD-1553, que utiliza cableado óptico en lugar de eléctrico, es conocida como MIL-STD-1773. La versión civil está definida en el estándar ARINC-429. Un sistema típico MIL-STD 1553 está compuesto de:
  - Un bus MIL-STD-1553B con redundancia doble
  - Un controlador de bus. En cualquier bus MIL-STD-1553, únicamente puede haber uno, es el encargado de iniciar toda comunicación de mensajes.
  - Un monitor de bus. No puede transmitir mensajes sobre el bus de datos. Se encarga de monitorizar y grabar las transacciones del bus, sin interferir en la operación del controlador de bus o los terminales remotos. Las transacciones del bus pueden ser almacenadas para un posterior análisis off-line. Otra funcionalidad es ser usado en conjunto con un controlador de bus auxiliar. Esto permite al controlador auxiliar estar en funcionamiento, incluso antes de tener que pasar a ser el controlador de bus activo.
  - Terminales remotos. Pueden ser usados para proporcionar una interfaz entre el bus de datos MIL-STD-1553B y un subsistema añadido, o como puente entre dos buses MIL-STD-1553B.

***Mision Computer* (MC).** Es el computador encargado de que se cumpla la misión establecida. Antes de salir a volar, al UAV se le cargan los datos específicos para la misión que ha de realizar. Algunos computadores necesitan datos específicos de misión para llevar a cabo su cometido, este computador se encarga de proporcionarlos.

***Servo Control Computer* (SCC).** Este computador es el encargado de controlar las superficies de control del avión cuando se realiza un vuelo por control remoto. Desde la estación de tierra, por medio de un *joystick*, se "vuela"el avión. Las órdenes para que esto sea posible, se reciben a través de este computador, el cual las distribuye al FCC.

***Sensors Computer* (SC).** El UAV lleva instalados sensores que informan del estado de diferentes variables que miden, estos valores pueden influir en el desarrollo de la misión. Por ejemplo, el sensor de creación de hielo en las alas, es necesario que se informe en el caso de existir hielo, para que desde tierra se active el protocolo necesario para que desaparezca, ya que sería peligroso volar en esas condiciones. A este computador pertenece el computador *Sensor EO Management Computer* para el que se está desarrollando este proyecto.

***General Systems Computer* (GSC).** Aquí se engloban los sistemas generales que necesita el avión para poder volar. Por ejemplo, el sistema de Combustible (ATA-28), el sistema eléctrico de los equipos (ATA-24 y ATA 29-20) y es el que alimenta a los computadores, el tren de aterrizaje (ATA-32)...

***Attack Computer* (AC).** Sistema de ataque a otros aviones. No todos los UAVs disponen de él, el ATLANTE, por ejemplo, carece de él.

***Aims Defence Computer* (ADC).** Es el sistema de contramedidas del avión. Va ligado al sistema AC del avión y sirve para defender al avión de posibles ataques enemigos.

PARTE V

CONCLUSIONES Y TRABAJOS FUTUROS

Capítulo V.1

Conclusiones

Resumen

En este capítulo se hace balance del trabajo realizado durante los meses que ha durado el proyecto. Partiendo de los objetivos iniciales se revisan cuales de ellos se han cumplido y cuales no, se revisan los resultados que se han obtenido, los problemas encontrados, las soluciones propuestas y las lecciones aprendidas.

La industria aeronáutica es una de las más punteras en desarrollos tecnológicos, actualmente el *boom* del sector pasa por el desarrollo de los aviones no tripulados (UAVs). En principio se están desarrollando para la aviación no comercial, debido a la posibilidad de evitar pérdidas de vidas humanas, el objetivo final es aplicarlos en la aviación comercial, aunque este sector es mucho más reticente a su implantación.

Debido a estos nuevos desarrollos y, en concreto, al nuevo desarrollo del UAV ATLANTE, primer UAV español, que se está llevando a cabo en EADS, surgió el proyecto que se ha desarrollado. En un principio, el proyecto se dividió en dos subproyectos, un primer subproyecto donde se han implementado y probado los algoritmos que comandan la cámara del sensor EO que portan los UAVs en la parte inferior del fuselaje; y un segundo subproyecto consistente en la implementación de un clasificador de las imágenes que captura la cámara del sensor EO.

La primera parte del proyecto se desarrolló en EADS, y consistía en implementar los algoritmos de geo-localización, que permite obtener la posición del objetivo al que está apuntando la cámara del sensor EO; y el algoritmo de geo-apuntamiento, que permite calcular la orientación de la cámara para apuntar a un determinado objetivo.

En un principio, se proporcionaron las especificaciones de los algoritmos y el trabajo consistía en implementarlos y someterlos a una batería de pruebas unitarias; para a continuación, plantear la integración de los algoritmos con el programa *Microsoft Flight Simulator*. Como en todo proyecto de *software*, surgieron problemas durante la implementación, los cuales obligaron a que se realizara un estudio exhaustivo de los algoritmos y del *background* que conllevaban. Este contratiempo retrasó el proyecto varios meses y fueron necesarias varias iteraciones de implementación y pruebas unitarias hasta dar con la solución correcta.

Una vez conseguida la implementación correcta, se pasó a integrar los algoritmos con el simulador de vuelo, *Microsoft Flight Simulator* en un banco de aviónica en EADS. Aquí, los problemas surgieron cuando se intentaba indicar al módulo FSUIPC el valor de las variables que se deseaban modificar. La documentación era muy pobre, con lo que el estudio de la librería y la posterior búsqueda de soluciones de implemetación retrasaron de nuevo el proyecto varias semanas. Una vez conseguido el objetivo, que el programa *Microsoft Flight Simulator* colocase el avión en las coordenadas que se le indicaban por medio del módulo FSUIPC de manera correcta, se pasó a cargar el programa desarrollado para que un avión diese vueltas realizando círculos alrededor de un edificio escogido previamente. Estas pruebas llevaron consigo el reajuste de la implementación y por tanto, la ejecución de toda la batería de pruebas unitarias en cada iteración.

Al final, se consiguió el objetivo marcado al inicio del proyecto, se desarrollaron los dos algoritmos con la funcionalidad pedida y las pruebas fueron pasadas satisfactoriamente.

Paralelamente, se ha desarrollado en colaboración con la Universidad Carlos III de Madrid, un segundo subproyecto, la implementación del clasificador de imágenes. El primer paso fue la lectura de artículos y la búsqueda de información complementaria para comprender qué eran y en qué consistían las SVMs, el algoritmo PCA y el algoritmo de *Complex-Log*; y cómo funcionaban los recursos electrónicos que precisaba utilizar, la *toolbox* de MATLAB y la librería LIBSVM.

En esta segunda parte, el primer problema surgió con la librería LIBSVM, ya que no es compatible con cualquier versión de MATLAB, para la realización del proyecto se ha utilizado la versión R2007b. Una vez resuelto, el siguiente obstáculo a salvar era conocer el funcionamiento de la librería LIBSVM. Como se ha comentado durante el proyecto, la documentación es muy escasa, con lo que se ha requerido un ejercicio de investigación bastante grande. Una vez comprendido el algoritmo que se deseaba implementar, no surgieron muchos problemas durante el desarrollo del mismo y la obtención de los primeros resultados. Para la obtención de los resultados que se muestran en la memoria, surgieron algunos problemas, ya que las ejecuciones duraban mucho tiempo y MATLAB, por su propia arquitectura utiliza muchos recursos del sistema, lo que imposibilitaba

el trabajo en paralelo.

El objetivo final era la implementación y evaluación de la SVM en distintas situaciones, principalmente para imágenes rotadas. Este estudio buscaba que se pudiera entrenar una máquina sin la necesidad de que participasen en el entrenamiento muestras rotadas.

Como se ha explicado los primeros resultados de imágenes crudas y con ruido han sido los esperados, cuanta más información se poseía en las muestras de entrenamiento, mejor era la clasificación de las muestras de test. También se ha podido comprobar que cuanto mayor es la cantidad de ruido que poseen las imágenes, la clasificación de las mismas empeora.

Tras ver los resultados obtenidos, en muchos casos es necesario evaluar si la disminución en el error medio de clasificación compensa el aumentar el número de SV y de características. Existen aplicaciones en las que el gasto computacional frente a la rapidez de clasificación a costa de perder precisión puede compensar. Éstas son decisiones de diseño que es necesario tomar para cada situación concreta.

Al final tal y como se ha mostrado en la memoria, los resultados obtenidos han sido los esperados. El error aumenta cuando las imágenes están distorsionadas por ruido o rotadas. Es bueno que existan muestras con estas distorsiones durante el entrenamiento porque mejoran el porcentaje de error, aunque es cierto que aumentan el número de vectores soporte necesarios.

En las pruebas con imágenes rotadas, el objetivo era evaluar cómo de bueno era el algoritmo de *Complex-Log*, para así poder, debido a la dificultad de poseer una base de datos representativa, plantear la posibilidad de utilizar imágenes sin rotar en el conjunto de entrenamiento y que la clasificación se realizase de manera óptima.

Como se ha visto en la sección de resultados, éstos no han sido satisfactorios. La imposibilidad de realizar muchas realizaciones en las simulaciones debido al alto tiempo computacional que tomaban para realizarse, y el hecho de no haber implementado en totalidad el algoritmo *Complex-Log* han sido las principales causas expuestas como explicación a los malos resultados obtenidos.

Para comprobar que realmente la implementación realizada inicialmente del algoritmo *Complex-Log* era insuficiente, se ha decidido realizarla por completo, y se ha demostrado que la FFT es la fase que permite a la SVM reconocer las imágenes como invariantes a rotaciones. Esto viene justificado por el desplazamiento que produce la primera fase y que se elimina al aplicar la FFT, ya que el módulo de la misma es invariante a desplazamientos.

En esta segunda parte, la mayor dificultad, ha sido comprender los algoritmos y conseguir realizar una clasificación básica utilizando la librería LIBSVM. Otro de los obstáculos que se ha tenido que salvar ha sido el alto tiempo computacional que requerían las simulaciones realizadas, esto ha frenado la realización del proyecto.

Para finalizar, se ha realizado un estudio de integración de los algoritmos. En primer lugar se ha realizado un estudio en cuanto a la cantidad de recursos computacionales, temporales y espaciales, que consumen los algoritmos implementados.

Sobre los algoritmos de geo-localización y geo-apuntamiento se ha determinado que los recursos computaciones que requieren son siempre constantes, con lo que la integración en el computador del UAV es posible. En cambio, el algoritmo de clasificación de imágenes los recursos computaciones que necesitan dependen de las dimensiones de las variables de entrada, el tamaño de la imagen a clasificar, de la matriz de características y de la matriz de vectores soporte. Las dimensiones que se manejan de estas variables de entrada son muy grandes, con lo que se ha determinado que únicamente sería posible embarcarlos si éstas son muy pequeñas. También se ha comentado la posibilidad de embarcarlo en un DSP, pero es una opción que habría que estudiar y valorar más en profundidad. Por todos estos motivos se ha determinado realizar todos los estudios posteriores contando con que esta funcionalidad estará implementada en la estación de tierra.

Como parte final del proyecto, se ha planteado, de manera teórica, un esquema de integración de los algoritmos dentro del computador de gestión del sensor EO del UAV. Esta integración se ha planteado de manera teórica, explicando los sistemas con los que se podría interactuar y cómo se realizarían las comunicaciones entre ellos.

Capítulo V.2

Trabajos Futuros

Resumen

En este capítulo, se exponen las líneas de trabajo futuro del proyecto que se ha desarrollado.

Durante el desarrollo de un proyecto siempre surgen nuevas ideas, líneas de investigación, ampliaciones... La mayoría de las veces, debido a que están fuera del ámbito del proyecto, quedan excluidas y olvidadas. En este capítulo, se hace una recopilación de todas ellas para futuras investigaciones.

La primera parte del proyecto, que se ha desarrollado en EADS, consiste en la implementación de los algoritmos de geo-apuntamiento y geo-localización y la posterior integración con el programa *Microsoft Flight Simulator*. Para este subproyecto se plantean las siguientes líneas futuras de trabajo:

- Desarrollar el ***software* embarcable 100 %**.
- **Realizar pruebas en un rig real**, teniendo interfaces reales con los equipos de UAV.
- **Integrar los algoritmos en un computador con un sensor EO**. Permitiría probar si realmente la cámara recibe las órdenes de manera correcta y si el nivel de precisión es suficiente.

Para la segunda parte del proyecto, la implementación del clasificador de imágenes, que se ha desarrollado en la UC3M, se plantean las siguientes líneas futuras de trabajo:

- **Imágenes en el espectro de infrarrojos**. La cámara también es capaz de captar imágenes en el espectro de infrarrojo, sería interesante tratar esas imágenes y evaluar la SVM.
- **Aislar las imágenes**. Para la realización de este proyecto se ha partido de la suposición de que la imagen a clasificar había sido aislada con anterioridad. Sería interesante realizar un estudio sobre las técnicas de aislamiento de imágenes, implementarlas y realizar una evaluación del proceso completo.
- **Extracción de características**. Aunque está comprobado que PCA es el algoritmo de extracción de características más sencillo y eficiente. Sería interesante evaluar otros algoritmos, como por ejemplo ICA.
- **Realizar pruebas con partes de imágenes**. Una posibilidad es que en la imagen, por la perspectiva, aparezca únicamente un trozo de la misma. Sería interesante plantear la posibilidad de realizar la evaluación de la máquina SVM con imágenes que sean parte de las imágenes reales.
- **Imágenes en color**. Una opción es trabajar con imágenes en color y comprobar si las prestaciones de la máquina mejoran con respecto a trabajar con imágenes en blanco y negro.
- **Fondo de las imágenes**. Se ha supuesto, que el fondo de las imágenes es negro y liso, existe la posibilidad de que las imágenes a clasificar se encuentren sobre un fondo no homogéneo. Habría que realizar un estudio para que la SVM recibiera esta información y clasificara correctamente independientemente del fondo de las imágenes.
- **Pruebas con imágenes escaladas y desplazadas**. Distinguiendo entre utilizar una rejilla lineal o logarítmica.
- **Pruebas con valores de iteraciones y número de imágenes mayor**. Esto serviría para conocer si realmente la dependencia con las imágenes y las muestras escogidas, es tan grande como se ha reflejado en los resultados obtenidos.
- **Banco de imágenes**. Sería interesante evaluar el clasificador con un banco de imágenes reales del entorno de desarrollo del proyecto. Esto no permitiría determinar la capacidad de generalización y eficacia de la SVM en la clasificación de las imágenes.

- **Integración en un DSP**. Para comprobar realmente la viabilidad de embarcar el algoritmo, sería interesante realizar una implementación en un DSP embarcable o lo más parecido, para constatar que se puede realizar la clasificación en tiempo real.

PARTE VI

APÉNDICES

Apéndice A

Presupuesto

Resumen

En este capítulo se muestra el presupuesto del proyecto. En él se especifican las horas de trabajo y el coste que supone la realización del proyecto para un cliente.

El primer paso de todo proyecto es la elaboración de un presupuesto, que especifique las horas de trabajo que exige y el coste que supone para el cliente su realización. Este presupuesto es necesario que se ajuste lo más posible a la realidad; ya que cualquier retraso en la planificación inicial, supone un gran impacto en el coste del proyecto y en la cuenta de resultados de la compañía que lo realiza.

La realización de un proyecto implica la ejecución de un conjunto de tareas relacionadas entre sí. Cada tarea del proyecto lleva asociado un tiempo de duración y unos recursos. En este capítulo, se van a detallar las tareas que se han realizado durante el proyecto, la duración y los recursos asociados a cada una de ellas. Todos estos recursos acarrean unos gastos que componen el presupuesto del proyecto.

A.1 Tareas realizadas durante el proyecto

A continuación, se enumeran y explican las tareas realizadas durante el proyecto:

- **Definición de los objetivos del proyecto.** Es una de las fases más importantes. Se han realizado distintas reuniones con el cliente, para fijar los objetivos del proyecto y la funcionalidad del producto que se desea crear. Ha de quedar todo perfectamente definido para que no se produzcan cambios de última hora.

- **Estudios preliminares del proyecto.** Para aceptar la realización del proyecto, se ha realizado un estudio exhaustivo sobre las tecnologías que se pueden utilizar para el desarrollo del mismo, el *hardware* y los recursos necesarios,... Con toda la información recopilada, se ha determinado si la realización del proyecto es viable, o no.

  A continuación se muestran las tareas que se han realizado en este proyecto:

  1. Definición de requisitos del sistema a implementar
  2. Lectura de papers y búsqueda de información sobre tipos de coordenadas y cambios entre ellas, proyecciones, librería FSUIPC y Microsoft Flight Simulator.
  3. Lectura de papers y búsqueda de información sobre clasificadores, PCA, SVM, SVM multiclase y *Complex-Log*.
  4. Estudio de ejemplos previos.
  5. Estudios sobre los lenguajes de programación que se van a utilizar.

- **Instalación del *software* y *hardware* necesario.** Todo proyecto implica la utilización de un *software* y un *hardware* específico. La instalación, en ciertas ocasiones, no es trivial; por lo que implica la dedicación de cierto tiempo a la realización de la misma. Las tareas realizadas en esta fase son:

  1. Instalación de Microsoft Visual Studio 6.0.
  2. Instalación de la librería LIBSVM y de la versión de MATLAB R2007b.
  3. Obtención de la base de datos COIL-100.

- **Diseño de la batería de pruebas.** Como se ha comentado en la memoria, es fundamental diseñar una buena batería de pruebas. Para este proyecto las pruebas van a ser unitarias y de integración en un RIG de EADS.

- **Implementación del proyecto.** Una vez realizados todos los estudios que aseguran la viabilidad del proyecto e instalado el *software*, hay que programar la aplicación. En esta fase del proyecto se producen retrasos debido a la complejidad que conlleva el desarrollo de aplicaciones.

  Las tareas iniciales que se han planificado son las que se muestran a continuación:

  1. Implementación del algoritmo de geo-apuntamiento.
  2. Implementación del algoritmo de geo-localización.
  3. Integración de la aplicación en el banco de aviónica.
  4. Implementación del clasificador de imágenes.
     a) Implementación del Preprocesado de las imágenes.
     b) Implementación del algoritmo de *Complex-Log*.
     c) Implementación de PCA.
     d) Implementación del entrenamiento y testeo de la SVM.

- **Realización de Pruebas, estadísticas del proyecto, revisión de errores y modificaciones.** En esta fase se ha sometido el proyecto a la batería de pruebas que se ha diseñado anteriormente. Se han obtenido resultados y realizado las estadísticas requeridas sobre la memoria utilizada por la aplicación y sobre el tiempo de ejecución de la misma. Las pruebas realizadas han sido descritas con precisión en el capítulo II.5 y III.6

  Durante la realización de las pruebas, se han encontrado errores en las aplicaciones; lo que ha provocado una replanificación del proyecto, búsqueda de soluciones, estudio de viabilidad de las mismas, cálculo del impacto en el proyecto, implementación de las soluciones y ejecución de la batería de pruebas a la nueva solución. Este proceso iterativo se ha repetido hasta que todas las pruebas han sido pasadas de manera satisfactoria.

- **Ejercicio teórico de integración.** Se ha realizado un ejercicio teórico de cómo se podría realizar una integración teórica del *software* desarrollado en un UAV real. Se han estudiado los sistemas con los que interactúa, los interfaces que posee, como se comunica con el resto del avión, ...

- **Documentación del proyecto.** Para finalizar el proyecto, es imprescindible crear un documento donde se incluyen los pasos a seguir para su realización, las tecnologías utilizadas, las pruebas realizadas con los resultados obtenidos y las variaciones que se han producido con respecto a la planificación inicial.

  La redacción del documento final ha pasado durante el proceso por varios borradores corregidos por los tutores y comentados con el autor del mismo.

Estas actividades se han realizado desde el 1 de Septiembre del 2009 hasta el 10 de Junio del 2010, día en que se realizará la presentación de este proyecto.

A.2 Análisis y valoración de recursos/costes

Para todas y cada una de las actividades descritas en la sección A.1, se ha efectuado una valoración de los recursos invertidos y así proceder a calcular el coste de cada actividad del proyecto.

Cada actividad tiene asociado un coste llamado coste directo. El valor del coste directo de una tarea está compuesto principalmente por los costes de los recursos humanos (RRHH) utilizados.

En las siguientes secciones se realiza un análisis y una valoración, tanto de los costes directos como indirectos del proyecto.

A.2.1 Costes directos

Los costes directos son los generados directamente por el proyecto, indispensables para la realización del mismo e imputables directamente a la realización de cada una de las tareas. Si el proyecto no se llevase a cabo, no existirían costes directos.

A.2.1.1 Material

En todos los proyectos es necesario utilizar material, ya sea informático o no, como por ejemplo cartuchos de impresora para imprimir las actas de las reuniones, los borradores, las presentaciones, los contratos con el cliente, los desplazamientos,...

A continuación se detallan los costes por material

- Lugar de trabajo: el proyecto se ha realizado entre la empresa EADS y el hogar del desarrollador. Se podría estimar un coste de 35€/hora por la electricidad, el agua y la limpieza durante los 11 meses de trabajo. En total 385€.
- Material de oficia: Dentro de este concepto se incluye todo el material desechable empleado en el proyecto, impresión de artículos, informes, cuadernos, bolígrafos,... El gasto total estimado es 116€.
- Equipo informático: Se estima el valor del equipo informático utilizado sobre 1525€. La impresora es un equipo que se va a utilizar fundamentalmente al final del proyecto, con lo que se estima que su valor es el 60 % sobre el total en concepto de amortización, con lo que obtiene 1200€. Este dato incluye los útiles de la impresora, como por ejemplo los cartuchos.
- Utilización del banco de pruebas de aviónica: Se estima que las horas invertidas en el banco han sido 300. Si el precio de alquiler del banco para su utilización en el proyecto es de 20€/hora, el coste total de su utilización es de 6000€.
- Licencias de *software*: Las licencias que se van a necesitar son: la de MATLAB, que tienen un coste de 2400€, la de Windows Vista que cuesta 200€, la del Microsoft Fligth Simulator 40€y la de la librería FSUIPC que se estima en 30€. El precio total asciende a 2670€.
- Conexión a internet: ADSL de Telefónica tiene un coste de 20€/mes, si se ha utilizado 11 meses, el gasto es de 220€.

- Impresión, encuadernación y copias del documento actual: el coste se estima en 120€.

Lugar de trabajo	385€
Material de oficia	116€
Equipo informático	1.200€
Utilización del banco de pruebas de aviónica	6.000€
Licencias de *software*	2670€
Conexión a internet	220€
Impresión, encuadernación y copias del documento actual	120€
TOTAL	10.711€

Tabla A.1: Costes directos imputables al proyecto.

A.2.1.2 Recursos Humanos

Según los datos del Colegio de Ingenieros de Telecomunicación (COIT), el sueldo por hora de un ingeniero es de 85€.

En la realización de este proyecto han participado tres personas, dos directores de proyecto y el desarrollador. Teniendo una carga del 20 % los tutores y un 80 % el desarrollador.

Las horas dedicadas al proyecto han sido de una media de 9 horas diarias. Si se computan 25 días al mes (además de los 20 días laborales, durante los días no laborables también se ha continuado el desarrollo), se obtiene
10meses*25 días/mes*9 horas/dia*85€/hora=191.250€

El salario del director de proyecto se estima, de forma general, como un 7 % del coste del material del proyecto, lo que suma 749,77€cada director de proyecto.

Sumando ambos costes se obtiene, 192.749,54€

A.2.2 Costes indirectos

Los costes indirectos son todos aquellos costes derivados de la actividad general de la empresa y que se distribuyen de forma proporcional entre los distintos proyectos que desarrolla. Toda empresa está formada por distintos departamentos que generan unos gastos independientes de la actividad o proyectos. A continuación se citan algunos ejemplos:

- **Dirección.** Son los puestos de dirección de la empresa; como por ejemplo, el Consejo de administración, Directores generales,...Se estima el coste en 196€.
- **Departamento de RRHH.** Son los responsables de la contratación de personal, de las nóminas, de los seguros sociales, de los beneficios sociales, de la formación del empleado,...Se estima el coste en 328€.
- **Departamento de Administración.** Se encargan de la facturación, de la contabilidad, de las relaciones con los bancos,...Se estima el coste en 163€.
- **Departamento Comercial.** La búsqueda de clientes y de proyectos corre a cargo de los empleados de este área. Se estima el coste en 232€.

- **Departamento de *Marketing*.** La publicidad, las presentaciones,... se realizan en este departamento. Se estima el coste en 329€.

La suma de todos los costes es 1.248€.

A.2.3 Costes totales

Los costes totales se calculan sumando los costes directos más los costes indirectos, tal y como muestra la tabla A.2. Si el proyecto se realiza para un cliente externo a la empresa, hay que añadir el beneficio industrial, un 15 % del coste total, y el IVA.

Material	10.711€
Recursos Humanos	192.749,54€
COSTES DIRECTOS	203.460,54€
COSTES INDIRECTOS	1.248€
TOTAL	204.708,54€
Beneficio industrial (15 %)	30.706,281€
SUBTOTAL	235.414,821€
IVA (16 %)	37.666,37136€
TOTAL	**273.081,19236€**

Tabla A.2: Tabla de los costes totales del proyecto.

Hay que tener en cuenta, que a partir del 1 de Julio, en lugar de aplicar un 16 % de IVA se deberá aplicar un 18 %.

Apéndice B

Acrónimos

Resumen

En este capítulo se listan los acrónimos utilizados en este documento.

Debido a la naturaleza del proyecto y a la empresa donde se ha desarrollado el mismo, es muy habitual el uso de acrónimos. Por este motivo, se han elaborado las listas B.1 y B.2. Cada una de éstas listas contiene los acrónimos pertenecientes a cada uno de los subproyectos elaborados.

Acrónimo	Inglés	Español
AC	Aircraft	Avión
ATLANTE		Avión Táctico de Largo Alcance No Tripulado Español
BIH	Bureau International de l'Heure	
CTRS	Conventional Terrestrial Reference System	Sistema de Referencia Convencional Terrestre
CTP	Conventional Terrestrial Pole	Polo Terrestre Convencional
EADS	European Aeronautics Defence and Space Company	Compañía Europea Aeronaútica Espacial de Defensa y Espacio
EASA	European Aviation Safety Agency	Agencia Europea de Seguridad Aérea
ECEF	Earth centered earth fixed	Centrado en la tierra, fijo en la tierra
ED xx	European Datum 19xx	
EO	Electro-optical	Electro-optico
FAA	Federal Aviation Administration	Administración Federal de Aviación
FCC	Flight Control Computer	Computador de control de vuelo
FS	Flight Simulator	Simulador de Vuelo
FoF	Friend or Foe	Amigo o enemigo
FSUIPC	Flight Simulator Universal Inter-Process Comunication	Comunicación entre procesos universal del simulador de vuelo
GCS	Ground Control Station	Estación de control de tierra
GPS	Global Positioning System	Sistema de posicionamiento Global
ISTAR	Intelligence, Surveillance, Target Acquisition and Reconnaissance	Inteligencia, Vigilancia, Adquisición de blancos y Reconocimiento
LLA	Latitude, Longitude and Altitude	Latitud, Longitud y Altura
LOS	Line Of Sigth	Línea de visión
MAS	Military Air Systems	Sistemas Aéreos Militares
MEDEVAC	MEDical EVACuation	Evacuación médica
MIDS-LVT	Multifunctional Information Distribution systems - Low Volume Terminal	
MISRA-C	Motor Industry Software Reliability Association C	
NAD xx	North American Datum 19xx	
NATO/OTAN	North Atlantic Treaty Organization	Organización del Tratado del Atlántico Norte
NED	North East Down	Norte Este y Sur
OSGB	Ordnance Survey Great Britain	
SATCOM	Satellite Communications	Comunicaciones por satélite
SDK	Software Development Kit	
STANAG	STANdarization AGreement	Acuerdos de estandarización
RAE		Real Academia Española
TGT	Target	Objetivo
UAS	Unmanned Aerial System	Sistema Aéreo No Tripulado
UAV/VANT	Unmanned Aerial Vehicle	Vehículo Aéreo No Tripulado
UCAV	Unmanned Combat Aerial Vehicles	Vehículo Aéreo de Combate No Tripulado
URAV	Unmanned Reconnaissance Aerial Vehicles	Vehículo Aéreo de Reconocimiento No Tripulado
VTOL	Vertical Take-off and Landing	Despegue y Aterrizaje Vertical
WGS xx	World Geodetic System 19xx	Sistema Geodético Mundial de 19xx

Tabla B.1: Lista de acrónimos (Subproyecto EADS).

Acrónimo	Inglés	Español
AvA	All vs All	Todos contra todos
CLM	Complex-Log Mapping	Mapa Logaritmo Complejo
COIL	Columbia Object Image Library	Librería de imágenes de Objetos de la universidad de Columbia
FFT	Fast Fourier Transform	Trasformada rápida de Fourier
MATLAB	MATrix LABoratory	Laboratorio de Matrices
OvA	One vs All	Uno contra todos
SVM	Support Vector Machine	Máquinas de vectores soporte
UC3M	Carlos III University of Madrid	Universidad Carlos III de Madrid

Tabla B.2: Lista de acrónimos (Subproyecto UC3M).

Bibliografía

[ANM09] Sammer A.Nene, Shree K. Nayar, and Hiroshi Murase. Columbia object image library (coil). technical report no. cucs-006-96, 2009.

[API05] *MSDN Library Visual Studio 2005.*, 2005.

[ATA10] Ata 100 chapter and section headings, 2010. http://www.s-techent.com/ATA100.htm.

[Chi02] John Joseph Chilenski. Software development under do-178b. 2002.

[COI10] Colegio oficial de ingenieros de telecomunicación (coit). 2010. www.coit.es.

[Cor09] EADS Corporate. Dvd promocional: Discover the world of eads, 2009.

[CS00] K. Crammer and Y. Singer. *On the learnability and design of output codes for multi-class problems. In Computational Learning Theory*. 2000.

[CS01] K. Crammer and Y. Singer. *On the algorithmic implementation of multiclass kernel-based vector machines. Journal of Machine Learning Research*. 2 edition, 2001.

[DD] Chris H.Q. Ding and Inna Dubchak. Multiclass proetin flod recognition using support vector machines and neural networks. Technical report, University of California.

[dDE09] Ministerio de Defensa Español. Monografías del sopt uas (unmanned aircraft system) sobre su integración en el espacio aéreo no segregado. 2009.

[Dep] Departamento de Informática y sistemas de la Universidad de las Palmas de Gran Canaria. *Transformaciones 2D*. http://www2.dis.ulpgc.es/ ii-fgc/Tema 3 - Transformaciones 2D.pdf.

[DS09] EADS División Defence and Security. Especificación de la aplicación. 2009.

[Duc07] Amalia Duch. Análisis de algoritmos. 2007.

[EAD10] EADS. Página oficial de eads, 2010. www.eads.com.

[FC01] Ignacio Alonso Fernández-Coppel. Localizaciones geográficas. las coordenadas geográficas y la proyeccion utm (universal transversa mercator). *Departamento de Ingeniería Agrícola y Forestal. Universidad de Valladolid*, 2001.

[FFT10] Curso de tratamiento digital de imágenes, 2010. http://www.diac.upm.es/acceso_profesores/asignaturas/tdi/tdi/transformadas/pdf/fourier3.pdf.

[for10] Foro kernel-machines.org, 2010. http://agbs.kyb.tuebingen.mpg.de/km/bb/forumdisplay.php?fid=7.

[FSU04a] *FSUIPC for Advanced Users.*, 2004.

[FSU04b] *FSUIPC for Programmers.*, 2004.

[FSU04c] *FSUIPC History.*, 2004.

[FSU04d] *FSUIPC User Guide.*, 2004.

[GWP01] Mohinder S. Grewal, Lawrence R. Weill, and Angus P.Andrews. *Global positioning systems, Inertial Navigation and Integration.* John Wiley & Sons, 2001.

[Hay99] Simon S. Haykin. *Neural networks : a comprehensive foundation*. Prentice Hall, 1999.

[HCL09] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. 2009.

[HL] Chih-Wei Hsu and Chih-Jen Lin. A comparision of methods for multi-class support vector machines. *National Taiwan University*.

[HS] Ellis Horowitz and Sartaj Sahni. *Fundamentals of computer algorithms*. Computer Science Press.

[Ima09] Base de imágenes de la coil-100, 2009. www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php.

[Kag91] Stoshi Kageyu. Augmented multi-layer perceptron for rotation-and scale invariant hand-wirtten numeral recognition. *Nagoya University, Japan*, 1991.

[Kle08] Herbert Klenk. System development framework document - sdf-f-0001. Technical Report Issue 1.2, EADS Defence & Security, October 2008.

[Mar08] Noelia López Martín. Reconocimiento de imágenes de baja resolución mediante clasificadores svm. Master's thesis, Universidad Carlos III de Madrid, 2008.

[Mat07] *Matlab Help r2007b.*, 2007.

[Mes07] Scott Messner. An overview of rtca do-178b. 2007.

[MG02] Andrés Marzal and Isabel Gracia. Introducción al análisis de algoritmos. 2002.

[MIS09] *Documentación del Master de Integración de Sistemas de Aeronaves.* EADS-CASA y Universidad Carlos III de Madrid, 2009.

[Mit97] Tom Michael Mitchell. *Machine learning*. McGraw-Hil, 1997.

[oDE98] Department of Defence EEUU. Multiplex application handbook mil-hdbk-1553a. 1998.

[P.S87] John P.Snyder. *Map projections - A working manual.* 1987.

[PV07] Massimiliano Pontil and Alessandro Verri. Support vector machines for 3-d object recognition. 2007.

[Reu] Fabián Reuter. Nociones de cartografía, proyecciones, sistemas de referencia y coordenadas en argentina. *Facultad de Ciencias Forestales*.

[RK04] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. 2004.

[San] Raúl Sánchez Santofimia. Reconocimiento en imágenes de baja resolución mediante clasificadores svm (máquinas de vectores soporte.). Master's thesis, Universidad Carlos III de Madrid.

[SBS98] Bernard Schölkopf, Christopher J.C. Burges, and Alexander J. Smola. Introduction to support vector learning. 1998.

[Sch00] Bernard Schölkopf. Statistical learning and kernel methods. 2000. http://research.microsoft.com/apps/pubs/default.aspx?id=69756.

[Sen10] Blog de desarrollo y defensa. 2010. http://desarrolloydefensa.blogspot.com/2009/09/colombia-adquiere-sensores.html.

[Smi02] Lindsay I Smith. A tutorial on principal components analysis. 2002.

[SS02] Bernhard Schölkopf and Alexander J. Smola. *Learning with kernels : support vector machines, regularization, optimization, and beyond*. MIT Press, 2002.

[Str00] Bjarne Stroustrup. *The C++ Programming Language.* Addison Wesley, special edition, 2000.

[svm10] Introduction to support vector machine (svm) models, 2010. http://www.dtreg.com/svm.htm.

[Too09] Toolbox de matlab par el procesado de imágenes., 2009. http://www.mathworks.com/access/helpdesk/help/toolbox/images/.

[Uni09] Columbia University. Página web de computer vision laboratory, 2009. www1.cs.columbia.edu/CAVE/software.

[vDH97] Foley van Dann and Feiner Hughes. *Computer graphics. Principles & Graphics.* Addison-Wesley publishing company, 2 edition, July 1997.

[Vel] Eduardo R. Lemus Velázquez. Coordenadas homogéneas. *Universidad Panamericana*. http://www.mx.up.mx/files_uploads/16226_EduardoLemus_CoordenadasHomogeneas.pdf.

[VL63] V Vapnik and A Lerner. *Pattern recognition using generalized portrait method*. 1963.

[VOP+02] M.A. Vicente, O.Reinoso, C. Pérez, J.A. Sabater, and J.A. Azorín. Reconocimiento de objetos 3d mediante análisis pca. 2002.

[WGS00] *Department of Defence World Geodetic System 84. Its definition and relationShip with Local Geodetic System (Technical Report)*. National Imagery and Mapping Agency, third edition, Enero 2000. www.globus.org/.

[WW99] J. Weston and C. Watkins. *Multi-class support vector machines. In Proceedings of the European Symposium on Artificial Neural Networks (ESANN)*. 1999.

[ZFM] Arkaitz Zubiaga, Víctor Fresno, and Raquel Martínez. Comparativa de aproximaciones a svm semisupervisado multiclase para clasificación de páginas web. Technical report, Universidad Nacional de Educación a Distancia.